UNIVERSIDADE FEDERAL DO CEARÁ

CAMPUS DE RUSSAS

CURSO DE GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

PEDRO LUCAS DA COSTA VIDAL

O JOGO DA COLORAÇÃO GULOSA: EXPLORANDO A PSPACE-COMPLETUDE COM 3 CORES

RUSSAS

2025

PEDRO LUCAS DA COSTA VIDAL

O JOGO DA COLORAÇÃO GULOSA: EXPLORANDO A PSPACE-COMPLETUDE COM 3 CORES

Trabalho de Conclusão de Curso apresentado ao Curso de Graduação em Ciência da Computação do Campus de Russas da Universidade Federal do Ceará, como requisito parcial à obtenção do grau de bacharel em Ciência da Computação.

Orientador: Prof. Dr. Eurinardo Rodrigues Costa.

RUSSAS

2025

Dados Internacionais de Catalogação na Publicação
Universidade Federal do Ceará
Sistema de Bibliotecas
Gerada automaticamente pelo módulo Catalog, mediante os dados fornecidos pelo(a) autor(a)

V692j Vidal, Pedro Lucas Da Costa.
O jogo da coloração gulosa : explorando a PSPACE-completude com 3 cores / Pedro Lucas Da Costa Vidal. – 2025.
61 f. : il. color.

Trabalho de Conclusão de Curso (graduação) – Universidade Federal do Ceará, Campus de Russas, Curso de Ciência da Computação, Russas, 2025.
Orientação: Prof. Dr. Eurinardo Rodrigues Costa.

1. Jogos de coloração. 2. PSPACE-completude. 3. Coloração gulosa. 4. Teoria dos jogos. I. Título.
CDD 005

PEDRO LUCAS DA COSTA VIDAL

O JOGO DA COLORAÇÃO GULOSA: EXPLORANDO A PSPACE-COMPLETUDE COM 3 CORES

Trabalho de Conclusão de Curso apresentado ao Curso de Graduação em Ciência da Computação do Campus de Russas da Universidade Federal do Ceará, como requisito parcial à obtenção do grau de bacharel em Ciência da Computação.

Aprovada em: 14/03/2025.

BANCA EXAMINADORA

Prof. Dr. Eurinardo Rodrigues Costa (Orientador)
Universidade Federal do Ceará (UFC)

Prof. Dr. Pablo Luiz Braga Soares
Universidade Federal do Ceará (UFC)

Prof. Dr. Rennan Ferreira Dantas
Universidade Federal do Ceará (UFC)

À minha família, por todo o esforço, amor e paciência ao longo da minha graduação. Mãe e pai, obrigado por serem pessoas tão batalhadoras e nunca deixarem de acreditar no meu potencial.

AGRADECIMENTOS

Aos meus pais, Elineuta e Pedro, por todo o esforço para criar a mim e meus irmãos com a melhor qualidade de vida possível ao longo de todos esses anos. Obrigado, mãe, por se esforçar tanto todos os dias e me ajudar sempre que possível. Obrigado, pai, por, mesmo trabalhando longe de casa, ainda apoiar muito sua família. Agradeço, também, aos meus irmãos mais novos, Diego e Sarah, pela companhia.

Agradeço muito ao meu orientador, Prof. Dr. Eurinardo, pela paciência e pelo apoio na realização deste trabalho.

RESUMO

Na área de Computação, existem jogos e enigmas matemáticos bastante interessantes para serem analisados. Como exemplo, temos os Jogos de Coloração em Grafos, nos quais alguns desses jogos consistem em colorir os vértices de um grafo seguindo um conjunto de regras predefinidas. Tais jogos levantam algumas questões que podem ser exploradas a respeito de sua complexidade computacional. Neste trabalho, focamos no *Jogo da Coloração Gulosa*, que se resume a um jogo de turnos, onde dois jogadores, que são Alice e Bob, fazem jogadas colorindo os vértices de um determinado grafo de modo que vértices vizinhos tenham cores distintas. Consideramos que as cores são números inteiros não negativos e que, quando o jogador escolher um vértice para colorir, a menor cor disponível deve ser utilizada. O conjunto de cores Cr do jogo é finito e Alice vence o jogo se for possível colorir todos os vértices do grafo com apenas as cores de Cr. O jogador Bob vence o jogo caso consiga forçar o uso de mais cores do que há disponível em Cr. O *Jogo da Coloração Gulosa* foi criado em 2013 por Havet e Zhu, que mostraram resultados acerca de alguns parâmetros referentes ao jogo. Em 2020, Costa *et al.* mostraram a PSPACE-completude do jogo na versão em que Alice inicia, porém, com um número grande de cores em Cr, mesmo para instâncias pequenas do problema. Neste trabalho, exploramos reduções polinomiais de problemas já conhecidos (POS-CNF e POS-CNF-6) para grafos, buscando estratégias que comprovem a complexidade com 3 cores para a versão que Bob inicia o *Jogo da Coloração Gulosa*. Identificamos na redução do problema POS-CNF, através do Lema 5.4.1, que prova que ciclos com tamanho $n \geq 5$ exigem 3 cores no jogo guloso, uma adversidade, na qual, para cláusulas longas, Bob sempre venceria seguindo uma certa estratégia. Para a redução do POS-CNF-6 e para grafos água-viva (grafos que possuem um C_6 induzido e subgrafos P_3 ligados ao C_6), identificamos possíveis configurações de jogo problemáticas quando Alice valora literais vizinhos numa cláusula. Implementamos uma versão jogável do *Jogo da Coloração Gulosa* (jogador vs Inteligência Artificial) utilizando o algoritmo MiniMax, que simula todas as jogadas possíveis e é útil para estudar configurações mais complexas. Concluímos que a PSPACE-completude com 3 cores permanece em aberto, mas nosso trabalho avança ao identificar configurações críticas em reduções polinomiais (cláusulas longas, literais vizinhos), estabelecer limites teóricos com o Lema 5.4.1 e prover ferramentas práticas (implementação do MiniMax e jogo interativo) para testes futuros.

Palavras-chave: jogos de coloração; PSPACE-completude; coloração gulosa; teoria dos jogos.

ABSTRACT

In the field of Computer Science, there are mathematical games and puzzles of great interest to be analyzed. As an example, we have Graph Coloring Games, where some of these games involve coloring the vertices of a graph following predefined rules. These games raise questions that can be explored regarding their computational complexity. In this work, we focus on the *Greedy Coloring Game*, a turn-based game where two players, Alice and Bob, take turns coloring the vertices of a graph such that adjacent vertices have distinct colors. The colors are non-negative integers, and when a player chooses a vertex to color, the smallest available color must be used. The set of colors Cr is finite, and Alice wins if all vertices are colored using only the colors in Cr. Bob wins if he forces the use of more colors than those available in Cr. The *Greedy Coloring Game* was introduced in 2013 by Havet and Zhu, who presented results on game-related parameters. In 2020, Costa *et al.* proved the PSPACE-completeness of the game in the version where Alice starts, but with a large number of colors in Cr, even for small problem instances. In this work, we explore polynomial reductions of known problems (POS-CNF and POS-CNF-6) to graphs, seeking strategies to prove the complexity of the 3-color version where Bob initiates the *Greedy Coloring Game*. Through Lemma 5.4.1, which proves that cycles of size $n \geq 5$ require 3 colors in the greedy game, we identified a challenge in the POS-CNF reduction: for long clauses, Bob can always win by following a specific strategy. For the POS-CNF-6 reduction and *água-viva* graphs (graphs containing an induced C_6 and P_3 subgraphs attached to the C_6), we identified problematic game configurations when Alice assigns values to adjacent literals in a clause. We implemented a playable version of the *Greedy Coloring Game* (player vs Artificial Intelligence) using the MiniMax algorithm, which simulates all possible moves and aids in studying complex configurations. We conclude that the PSPACE-completeness with 3 colors remains open. However, our work advances the field by identifying critical configurations in polynomial reductions (long clauses, adjacent literals), establishing theoretical bounds with Lemma 5.4.1, and providing practical tools (MiniMax implementation and interactive game) for future testing.

Keywords: graph coloring games; PSPACE-completeness; greedy coloring; game theory

LISTA DE FIGURAS

LISTA DE CÓDIGOS-FONTE

SUMÁRIO

1 INTRODUÇÃO

Na Computação, existem muitos jogos e enigmas matemáticos que trazem consigo muitas perguntas a serem respondidas do ponto de vista computacional. Dentre esses jogos, existem os de Coloração em Grafos, nos quais é possível colorir os vértices e as arestas de um determinado grafo, a depender das regras do jogo.

É possível encontrar vários tipos de coloração nos vértices de grafos. Como exemplo, tem-se a *Coloração Equitativa de Vértices*, a *Coloração Circular de Vértices* e a *Coloração Acíclica de Vértices*. Do mesmo modo, existem também diversos tipos de coloração de arestas (FORMANOWICZ; TANAŚ, 2012). Neste trabalho, focaremos na *Coloração Própria de Vértices*, que consiste em colorir os vértices do grafo de forma que, para cada vértice, aplica-se a regra de que o mesmo não pode ter a mesma cor que seus vizinhos.

A coloração de grafos tem muitas aplicações úteis no mundo real. Segundo (THADANI *et al.*, 2022, p. 01, tradução nossa), "Com a ajuda dos conceitos de coloração de grafos, podemos resolver diversos problemas de organização de horários, problemas de agendamento, problemas de alocação de tarefas, problemas de formação de equipes, etc." [1]. O artigo cita um exemplo para a aplicação em problemas de agendamento, que envolvia seis estações de radiofrequência. O objetivo era encontrar o número mínimo de diferentes canais de frequência para as seis estações, de forma que duas estações possam usar o mesmo canal caso estejam localizadas a mais de 150 milhas (aprox. 241 km) uma da outra. Utilizando-se tabelas e conceitos da coloração de vértices, foi concluído que, para as estações evitarem interferência entre si, são necessários, pelo menos, três canais de frequência diferentes (THADANI *et al.*, 2022).

Antes de falar sobre o *Jogo da Coloração Gulosa*, é importante descrever o funcionamento do jogo original ao qual o *Jogo Coloração Gulosa* toma como base e que foi descrito por (GARDNER, 1981 apud COSTA *et al.*, 2020, p. 1) na coluna "*Mathematical Games*" da revista *Scientific American*, que é o *Jogo da Coloração*.

O funcionamento do *Jogo da Coloração* se resume a, basicamente, um jogo de turnos, onde os jogadores, que são Alice e Bob, fazem jogadas colorindo os vértices de um determinado grafo com uma cor disponível de um conjunto predefinido Cr. As cores podem ser representadas por números inteiros não negativos. Antes de colorir um vértice com uma determinada cor, é necessário verificar a cor de seus vértices vizinhos, porque, pelas regras do

[1] "With the help of graph coloring concepts, we can solve various timetabling problems, scheduling problems, job allocation problems, team building problems, etc."

jogo, dois vértices vizinhos não podem compartilhar a mesma cor. Desde sua criação até então, o jogo foi bastante estudado e vários resultados foram obtidos acerca de limites de parâmetros do jogo em algumas classes de grafos. Entretanto, o primeiro resultado de complexidade foi obtido quase trinta anos depois da criação do jogo, em (COSTA *et al.*, 2020), que mostrou a PSPACE-completude do jogo.

Em (HAVET; ZHU, 2013), foi criado o *Jogo da Coloração Gulosa*, que é o foco deste trabalho. O jogo pode ser definido, basicamente, como o Jogo da Coloração normal, porém com uma nova regra aplicada. Nele, no momento em que o jogador escolhe a cor em Cr, aplica-se a regra de que somente a menor cor disponível poderá ser escolhida. Ou seja, a menor cor do conjunto que não esteja sendo utilizada pelos vizinhos do vértice a ser colorido.

Alguns resultados interessantes sobre a PSPACE-completude já foram encontrados em algumas variações do jogo. Por exemplo, a variação do *Jogo da Coloração Gulosa* em que Alice inicia o jogo foi mostrada ser PSPACE-completo em (COSTA *et al.*, 2020). Recentemente, em (LIMA *et al.*, 2023), cinco variações do Jogo da Coloração Gulosa foram mostradas serem PSPACE-completos. As versões de (LIMA *et al.*, 2023) seguem de Alice ou Bob iniciar o jogo, e a regra de que os vértices escolhidos devem ser vizinhos de outro vértice já colorido (quando existir). Porém, em todos estes resultados de PSPACE-completude, o número de cores nas demonstrações é grande, mesmo para instâncias pequenas (mais de 40 cores para um tipo de versão, por exemplo). Este trabalho tem como objetivo explorar a PSPACE-completude com apenas três cores no *Jogo da Coloração Gulosa*.

O texto do trabalho está organizado como se segue. No Capítulo 2, detalhamos o Objetivo Geral a qual este trabalho se propõe, além de também listar os objetivos específicos, que são necessários para se conquistar o objetivo principal.

No Capítulo 3, temos a fundamentação teórica necessária para um bom entendimento deste trabalho. Primeiro, vemos conceitos básicos de Teoria dos Grafos, Classe PSPACE e Algoritmo MiniMax. Em seguida, damos mais foco aos Jogos de Coloração, onde, primeiramente, fazemos uma descrição mais formal do *Jogo da Coloração*, apresentando o funcionamento de suas regras, mostrando algumas possíveis jogadas e também analisando possíveis estratégias que os jogadores podem utilizar. Posteriormente, fazemos o mesmo para o *Jogo da Coloração Gulosa*, comparando o seu funcionamento com o jogo original e vendo se as mesmas estratégias funcionam nos dois jogos.

O Capítulo 4 resume os três principais trabalhos acerca do Jogo da Coloração Gulosa.

No Capítulo 5, mostramos a metodologia empregada neste trabalho. Apresentamos três tentativas de redução polinomial com o uso dos problemas POS-CNF, POS-CNF-6 e grafos *águas-vivas* (nomeados assim, neste trabalho). Mostramos, também, um lema relacionado a ciclos, o qual acreditamos que possa ser utilizado para obter resultados em grafos *cactus*. Para finalizar o capítulo, mostramos uma implementação do algoritmo MiniMax e uma versão jogável do *Jogo da Coloração Gulosa*, que pode ser jogado contra uma Inteligência Artificial.

No Capítulo 6, discutimos sobre os resultados obtidos e sobre nossas contribuições para a resolução do problema central. No Capítulo 7, último capítulo, expomos nossas conclusões e trabalhos futuros.

2 OBJETIVOS

Neste capítulo, veremos os principais objetivos que este trabalho busca alcançar. Definir tais objetivos é de grande importância para ajudar a aumentar a credibilidade do trabalho e a deixar mais claro para o leitor o que pretendemos alcançar com esta pesquisa. A começar do nosso **Objetivo Geral**, damos uma breve descrição da ideia central do trabalho. Em seguida, serão listados os **Objetivos Específicos**, que são uma fragmentação da ideia principal e, juntos, servem de apoio para alcançar o objetivo geral proposto.

2.1 Objetivo Geral

Investigar e provar, através de análises e estratégias, a PSPACE-completude do *Jogo da Coloração Gulosa* e de suas variações quando analisada com apenas 3 cores.

2.2 Objetivos Específicos

- Revisar conceitos básicos de Teoria dos Grafos e de Complexidade Computacional;
- Analisar diferentes variações do *Jogo da Coloração Gulosa* e seu funcionamento;
- Verificar resultados sobre a PSPACE-completude do *Jogo da Coloração Gulosa* obtidos de outros trabalhos relacionados;
- Aplicar reduções polinomiais para construção de grafos a partir de fórmulas lógicas, pensando em estratégias vencedoras para cada um dos jogadores (Alice e Bob) e analisar os resultados;
- Utilizar de métodos computacionais para casos em que a abstração dos grafos e fórmulas são muito difíceis aos olhos humanos.

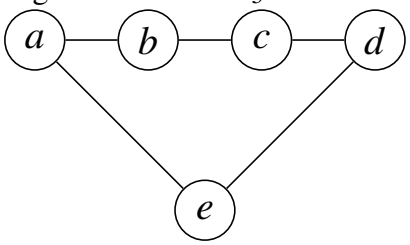
3 FUNDAMENTAÇÃO TEÓRICA

Neste capítulo, serão introduzidos os conceitos básicos teóricos fundamentais para o desenvolvimento deste trabalho, começando pela Teoria dos Grafos, Classe PSPACE e Algoritmo MiniMax. Esses conceitos são essenciais para compreender o funcionamento dos Jogos de Coloração. Em seguida, será discutido o que são os Jogos de Coloração, detalhando suas principais variações e apresentando definições que servirão de base para a compreensão do *Jogo da Coloração Gulosa*.

3.1 Teoria dos Grafos

Um grafo simples (ou apenas grafo) é um par $G = (V,E)$, onde V (ou $V(G)$) é um conjunto finito (chamado de vértices) e E (ou $E(G)$) é um conjunto de pares não ordenados de V (chamado de arestas). Por simplicidade, escrevemos uma aresta $\{u,v\}$ apenas por uv. Um grafo pode ser desenhado num plano, onde os vértices são círculos e as arestas são arcos ligando os círculos. Veja na Figura 1 o grafo chamado C_5 que possui o conjunto de vértices $\{a,b,c,d,e\}$ e conjunto de arestas $\{ab,bc,cd,de,ae\}$. Podemos escrever ainda $C_5 = (\ \{a,b,c,d,e\},\ \{ab,bc,cd,de,ae\})$.

Figura 1 – Grafo C_5



Fonte: elaborado pelo autor.

Dizemos que um grafo $H = (V',E')$ é *subgrafo* de um grafo $G = (V,E)$ se $V' \subseteq V$ e $E' \subseteq E$. Por exemplo, note que o grafo $P_5 = (\ \{a,b,c,d,e\},\{ab,bc,cd,de\})$ é subgrafo de C_5 (temos que P_5 é o próprio C_5 com a aresta ae removida). Caso H seja subgrafo de G, dizemos que H é um *subgrafo induzido* de G se, para cada par de vértices $u,v \in V(H)$, temos que $uv \in E(H)$ se e somente se $uv \in E(G)$. Note que P_5 não é subgrafo induzido de C_5, uma vez que $ae \notin E(P_5)$, mas $ae \in E(C_5)$.

Um *caminho simples* (ou caminho) num grafo simples G é uma sequência de vértices distintos $v_1, v_2, \cdots v_k$ de G, onde $\{v_i, v_{i+1}\} \in E(G)$ para $i = 1, \cdots, k-1$. Denotamos por P_k o

caminho que possui k vértices. No parágrafo anterior, descrevemos o P_5.

Um *ciclo simples* (ou somente *ciclo*) é um caminho com, pelo menos, três vértices $v_1, v_2, \cdots, v_k$ de G, onde $\{v_1, v_k\} \in E(G)$. Denotamos por C_k o ciclo que possui k vértices. Anteriormente, descrevemos o C_5 e na Figura 1 temos seu desenho.

Uma *k-coloração própria* ($k \in \mathbb{Z}$) é uma função $f : \{1, \cdots, k\} \to V$, onde $f(u) \neq f(v)$ se $\{u, v\} \in E$. Se existir uma k-coloração própria para um grafo, dizemos que ele é *k-colorível*.

Denotamos por $\chi(G)$ o *número cromático* de um grafo G, que é o menor $k \in \mathbb{Z}$ positivo tal que G é k-colorível. Tomando o C_5 da Figura 1 como exemplo, podemos colori-lo do seguinte modo: $f(a) = 1$, $f(b) = 2$, $f(c) = 1$, $f(d) = 2$ e $f(e) = 3$. Não temos como colorir C_5 com apenas duas cores, pois, como teríamos que colorir alternando duas cores seguindo suas arestas, para o último vértice, não podemos alternar, uma vez que ele estará ligado a dois vértices com duas cores distintas. Como C_5 não é 2-colorível, mas é 3-colorivel, temos que $\chi(C_5) = 3$.

Para consulta de outras definições e resultados em Teoria dos Grafos recomendamos (BONDY; MURTY, 2008).

3.2 Classe PSPACE e problema PSPACE-completo

Um problema de decisão é um problema que responde uma pergunta com SIM ou NÃO a cerca de uma determinada instância, que pode ser um *grafo*, *vetor*, *número*, etc (BONDY; MURTY, 2008, p. 175).

A Classe PSPACE é a classe de problemas de decisão que possuem algoritmos em espaço polinomial que os resolvem. Ou seja, um problema pertence a PSPACE se existe um algoritmo que o resolva com um tamanho de memória polinomial no tamanho da instância (SIPSER, 2012, p. 336).

Uma *redução polinomial* de um problema de decisão A para uma problema de decisão B é uma função f (algoritmo) que recebe uma instância w de A e retorna uma instância $f(w)$ de B em tempo polinomial, de modo que w responde *SIM* para A se e somente se $f(w)$ responde *SIM* para B.

Um problema de decisão é PSPACE-difícil se todos os problemas em PSPACE se reduzem polinomialmente a ele. Um problema é PSPACE-completo se pertence a PSPACE e é PSPACE-difícil. Neste trabalho, apresentamos algumas tentativas de reduções polinomiais, que estão descritas adiante no texto. Para um estudo mais aprofundado sobre o tema de complexidade computacional, recomendamos a leitura do livro (SIPSER, 2012).

3.3 Algoritmo MiniMax

O algoritmo *MiniMax* é um algoritmo geralmente utilizado para jogos baseados em turnos com dois jogadores (apesar de poder ser facilmente estendível para mais de dois jogadores em certos jogos) que gera uma árvore com todas as possibilidades de jogadas. Apesar de ter uma complexidade de tempo exponencial, o algoritmo pode ser utilizado para entradas que, embora sejam computacionalmente pequenas, são razoavelmente grandes para uma abstração humana. A seguir, vemos a recursão que deriva o algoritmo MiniMax, retirada do livro (RUSSELL; NORVIG, 2009, p. 164), em que s é uma configuração de jogo, AÇÕES são as jogadas possíveis a partir da configuração vigente e UTILIDADE é o valor de uma configuração de jogo (vitória, derrota ou empate, para fim de jogo, por exemplo).

$$\text{MINIMAX}(s) = \begin{cases} \text{UTILIDADE}(s) & \text{, se } s \text{ é fim de jogo} \\ \max_{a \in \text{AÇÕES}} \text{MINIMAX}(\text{RESULTADO}(s, a)) & \text{, se Jogador = MAX} \\ \min_{a \in \text{AÇÕES}} \text{MINIMAX}(\text{RESULTADO}(s, a)) & \text{, se Jogador = MIN} \end{cases}$$

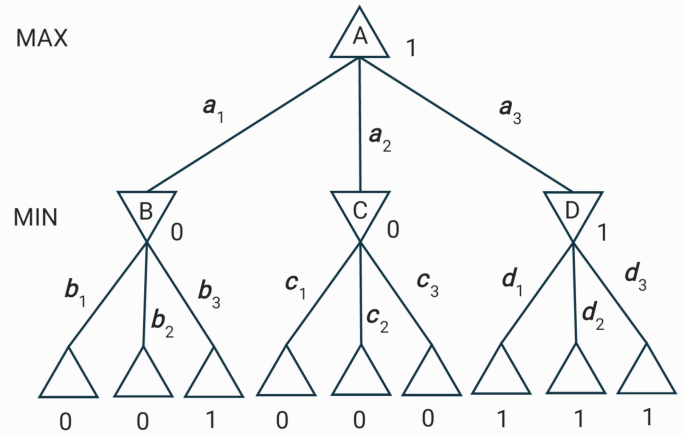
A partir da recursão anterior, podemos facilmente criar um algoritmo MiniMax para um jogo baseado no *Jogo da Coloração Gulosa*. No Capítulo 5, mostramos o pseudo-código do Algoritmo MiniMax para o JOGO DA COLORAÇÃO GULOSA COM 3 CORES e sua implementação na linguagem Python no Apêndice B.

Para um bom entendimento, considere uma possível árvore criada pela recursão MiniMax na Figura 2 para um jogo com duas jogadas. Consideramos um jogo de vitória-derrota, isto é, não temos empate. Os nós $\triangle$ são referentes ao jogador MAX e os nós $\triangledown$ são referentes ao jogador MIN. As jogadas possíveis dos nós A, B, C e D são $\{\text{a}_1,\text{a}_2,\text{a}_3\}$, $\{b_1, b_2, b_3\}$, $\{c_1, c_2, c_3\}$ e $\{d_1, d_2, d_3\}$, respectivamente. A UTILIDADE é 1 para vitória de MAX e 0 para sua derrota. Para o nó B, que é nó MIN, temos que sua UTILIDADE será 0, pois b_1 e b_2 levam a fim de jogo com UTILIDADE 0 (derrota de MAX e vitória para MIN). Já para o nó C (resp. D) temos UTILIDADE 0 (resp. 1) uma vez que em todas suas jogadas resultam em UTILIDADE 0 (resp. 1). Ao final, temos a vitória de MAX, uma vez que o filho D da raiz A possui UTILIDADE 1.

3.4 Parâmetros dos jogos de coloração

Para um bom entendimento acerca dos resultados dos jogos, nesta seção falaremos sobre as definições dos principais parâmetros dos jogos de coloração:

Figura 2 – Exemplo de árvore MiniMax



Fonte: elaborado pelo autor.

Número cromático de jogo $\chi_g(G)$

menor número de cores que Alice possui uma estratégia vencedora no Jogo da Coloração no grafo G.

Número guloso de jogo $\Gamma_g(G)$

menor número de cores que Alice possui uma estratégia vencedora no Jogo da Coloração Gulosa no grafo G.

3.5 O Jogo da Coloração

Nesta seção, daremos uma explicação mais detalhada sobre o *Jogo da Coloração*, seguindo (GARDNER, 1981 apud COSTA, 2022, p. 22).

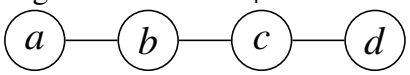
Dado um grafo simples $G = (V,E)$ e um conjunto Cr de cores com k elementos, onde tais cores são representadas por inteiros não negativos, o *Jogo da Coloração* é composto por dois jogadores, Alice e Bob, que alternam suas jogadas em cada turno. Abaixo, temos uma breve descrição das principais regras do jogo:

1. normalmente, tanto Alice quanto Bob podem dar início ao jogo, mas, para este trabalho, focaremos no caso onde Bob inicia;
2. como o jogo é de turnos e em cada turno é possível o jogador fazer uma jogada, a jogada é definida pelo jogador (Alice ou Bob) escolher um vértice v qualquer em G que não esteja colorido e, depois, colorir v com uma cor disponível em Cr, de forma que v não tenha cor igual a de seus vizinhos;
3. Alice é declarada vencedora do jogo quando todos os vértices de G forem coloridos com as cores de Cr. Caso não seja possível colorir todos os vértices apenas com as cores de Cr,

Bob sai como vencedor.

Para ficar ainda mais claro os conceitos do jogo, serão utilizados exemplos de jogadas e resultados. Veja na Figura 3 o grafo P_4 utilizado nos exemplos.

Figura 3 – Grafo P_4



Fonte: elaborado pelo autor.

Considerando o conjunto $Cr = \{1,2\}$, podemos atribuir a cor verde ao número 1 e a cor vermelha ao número 2, para facilitar o entendimento. Para definir as jogadas, seguiremos as notações criadas em (COSTA, 2022):

$A(v,c)$: jogada que *Alice colore um vértice* $v \in V(G)$ *com a cor escolhida* $c \in Cr$

$B(v,c)$: jogada que *Bob colore um vértice* $v \in V(G)$ *com a cor* $c \in Cr$

3.5.1 Exemplos em que Alice inicia o jogo

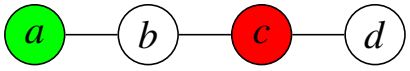
Apenas para exemplificar, nesta subseção, mostramos algumas jogadas possíveis quando a Alice inicia o jogo, pois, no caso especial do grafo P_4 da Figura 3, Bob consegue adotar uma estratégia interessante.

3.5.1.1 Rodada 1: Alice inicia colorindo um vértice extremo do P_4

- Turno 1: $A(a, \text{verde})$
- Turno 2: $B(c, \text{vermelho})$
- Turno 3: o vértice b não pode ser colorido, pois seus vizinhos, a e c, já estão coloridos com as cores verde e vermelho, respectivamente. Portanto, Bob é o vencedor dessa rodada.

Veja na Figura 4 o grafo resultante da Rodada 1.

Figura 4 – Grafo P_4 resultante da Rodada 1 do *Jogo da Coloração*



Fonte: elaborado pelo autor.

É possível notar que, em grafos do tipo *caminho* que possuem, pelo menos, quatro vértices, e com o conjunto Cr contendo duas cores, Bob consegue uma estratégia vencedora ao "encurralar" um vértice que possui um vizinho já colorido, colorindo seu outro vizinho com uma

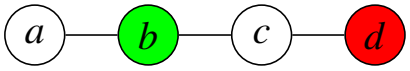
cor diferente.

Vamos analisar mais um exemplo, onde Alice escolhe um dos vértices centrais do P_4 para colorir, para verificar se a estratégia de Bob descrita anteriormente continua válida.

3.5.1.2 Rodada 2: Alice inicia colorindo um dos vértices centrais do P_4

- Turno 1: $A(b, \text{verde})$
- Turno 2: $B(d, \text{vermelho})$
- Turno c: o vértice c não pode ser colorido com as duas cores. Portanto, novamente, Bob sai como vencedor da rodada (veja a Figura 5).

Figura 5 – Grafo P_4 resultante da Rodada 2 do *Jogo da Coloração*



Fonte: elaborado pelo autor.

Sendo assim, como os casos para quando Alice escolher os demais vértices são análogos, podemos concluir que, para esse caso, Bob possui uma grande vantagem para vencer Alice, pois, independentemente do vértice ou cor que Alice escolher, Bob consegue aplicar sua estratégia.

3.5.2 Exemplos em que Bob inicia o jogo

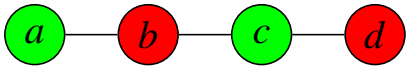
Diferentemente das rodadas anteriores, onde Alice iniciava o jogo, nesta subseção, estudaremos algumas jogadas de quando Bob inicia e analisaremos se há algumas outras estratégias possíveis.

3.5.2.1 Rodada 3: Bob começa colorindo uma das pontas do P_4

- Turno 1: $B(d, \text{vermelho})$
- Turno 2: $A(b, \text{vermelho})$
- Turno 3: $B(a, \text{verde})$
- Turno 4: $A(c, \text{verde})$ – Todos os vértices foram coloridos. Alice vence ao evitar a estratégia de Bob. Veja resultado da Rodada 3 na Figura 6.

Nesse caso, como Bob começou a rodada e escolheu o vértice d, que se encontra em uma das extremidades, e o pintou de vermelho, bastou com que Alice escolhesse um vértice

Figura 6 – Grafo P_4 resultante da Rodada 3 do *Jogo da Coloração*



Fonte: elaborado pelo autor.

com distância dois de d, que, nesse caso, foi o vértice b, e o colorisse com a mesma cor, visando evitar a estratégia de Bob. Alice pode usar a mesma estratégia para os casos onde Bob escolhe um vértice inicial diferente.

Com isso, concluímos que o P_4 é um grafo interessante no *Jogo da Coloração*. Bob consegue uma estratégia vencedora na variante que Alice inicia, enquanto Alice consegue uma estratégia vencedora na variante que Bob inicia.

3.6 O Jogo da Coloração Gulosa

Nesta seção, apresentamos o *Jogo da Coloração Gulosa*, que foi introduzindo por Havet e Zhu no artigo "*The game Grundy number of graphs*" (o número guloso de jogo em grafos) da revista *Journal of Combinatorial Optimization* (HAVET; ZHU, 2013).

Seja um grafo $G = (V, E)$ e um conjunto Cr de cores com inteiros positivos. Assim como descrito na Seção 3.5 ("*O Jogo da Coloração*"), Alice e Bob são os jogadores. Porém, no caso do *Jogo da Coloração Gulosa*, os jogadores alternam seus turnos escolhendo um vértice não colorido $v \in V(G)$ e o colorem de maneira *gulosa*. Em sequência, iremos definir as principais regras:

1. como na Seção 3.5, temos variações onde tanto Alice, quanto Bob, podem iniciar o jogo. Como já falado anteriormente, focaremos na variação onde Bob dá início;
2. em cada turno, a jogada é definida por Alice ou Bob escolherem um vértice $v \in V(G)$ que não esteja colorido e atribuírem à ele uma cor diferente da de seus vértices adjacentes;
3. ao atribuir uma cor à $v \in V(G)$, obrigatoriamente, é escolhido o menor número inteiro positivo de Cr que não esteja colorindo os vértices adjacentes de v;
4. no fim, Alice é declarada vencedora se colorir todos os vértices de G com as k cores de Cr. Caso contrário, Bob será o vencedor.

É possível notar que o *Jogo da Coloração Gulosa* se baseia muito no *Jogo da Coloração* (veja-o na Seção 3.5). Segundo (HAVET; ZHU, 2013, p. 2, tradução nossa), "De certa forma, este novo jogo é uma mistura do jogo da coloração e do jogo da marcação."[1]. Para

[1] "In some sense, this new game is a mixture of the colouring game and the marking game."

esclarecer ainda mais o funcionamento do jogo, vamos a alguns exemplos de jogadas. Para isso, considere novamente o grafo P_4 da Figura 3.

De modo semelhante à Seção 3.5, seguimos com as notações de jogadas de (COSTA, 2022):

$A(v)$: jogada que *Alice colore o vértice* $v \in V(G)$ *com a menor cor disponível*

$B(v)$: jogada que *Bob colore o vértice* $v \in V(G)$ *com a menor cor disponível*

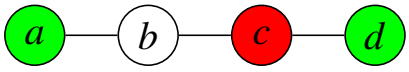
3.6.1 Jogo da Coloração Gulosa quando Alice inicia o jogo

Para esta subseção, vamos explorar algumas jogadas de quando a Alice inicia *Jogo da Coloração Gulosa* e ver possíveis estratégias.

3.6.1.1 Rodada 1 (variação gulosa): Alice inicia colorindo um vértice extremo do P4

- Turno 1: $A(a)$ [verde]
- Turno 2: $B(d)$ [verde]
- Turno 3: $A(c)$ [vermelho]
- Turno 4: Alice perde, pois não é possível colorir b com as cores de Cr. Veja o resultado da Rodada 1 na Figura 7.

Figura 7 – Grafo P_4 resultante da Rodada 1 do *Jogo da Coloração Gulosa*



Fonte: elaborado pelo autor.

É notório que a estratégia de Bob descrita na Seção 3.5.1 não funciona no *Jogo da Coloração Gulosa*, devido a sua regra de escolha de cor. No entanto, percebe-se que Bob tem uma nova estratégia vencedora caso Alice inicie colorindo um dos vértices da extremidade. Basta colorir a extremo oposto com a mesma cor, "encurralando" os vértices centrais. Para evitar isso, é evidente que Alice deve começar colorindo um dos vértices centrais do P_4.

3.6.2 Jogo da Coloração Gulosa quando Bob inicia o jogo

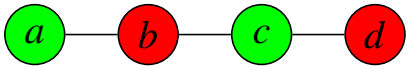
Agora, vamos explorar algumas jogadas de quando Bob dá início ao jogo. Como não faz sentido Bob começar colorindo nos vértices centrais b e c, pois isso seria vantajoso apenas

para Alice, vamos analisar as jogadas de quando Bob inicia colorindo uma das extremidades (vértice a ou d).

3.6.2.1 Rodada 2 (variação gulosa): Bob inicia colorindo um vértice extremo do P4

- Turno 1: $B(a)$ [verde]
- Turno 2: $A(c)$ [verde]
- Turno 3: $B(b)$ [vermelho]
- Turno 4: $A(d)$ [vermelho] – Todos os vértices foram coloridos. Portanto, Alice é a vencedora da rodada (Figura 8).

Figura 8 – Grafo P_4 resultante da Rodada 2 do *Jogo da Coloração Gulosa*



Fonte: elaborado pelo autor.

Observe que, para essa configuração de jogo (um grafo P_4 e $Cr = \{1,2\}$), o *Jogo da Coloração Gulosa* demostrou ser bastante favorável para Alice, já que ela vence nas duas variações. Diferentemente do que foi observado no *Jogo da Coloração* (Seção 3.5), onde Alice e Bob tinham uma estratégia a depender de quem iniciava o jogo.

Com os exemplos de jogadas de cada jogo vistos anteriormente, concluímos que, por mais que o *Jogo da Coloração* e o *Jogo da Coloração Gulosa* sejam parecidos a princípio, os resultados podem mudar bastante e, além disso, estratégias que funcionam em um, podem ou não funcionar no outro.

4 TRABALHOS RELACIONADOS

Neste capítulo, apresentamos alguns trabalhos relacionados aos *Jogos de Coloração* e seus resultados, com foco principal no versão gulosa. Veja a Seção 3.4 para uma melhor compreensão dos resultados.

4.1 The game grundy number of graphs

Em (HAVET; ZHU, 2013), foi introduzido o *Jogo da Coloração Gulosa*. Nele, os jogadores Alice e Bob alternam suas jogadas colorindo os vértices não coloridos de um determinado grafo $G = (V, E)$ com a menor cor (menor positivo inteiro) não usada pelos vizinhos do vértice escolhido.

O objetivo de Alice é minimizar o número de cores utilizadas no jogo, enquanto o de Bob é maximizar esse número. O trabalho ainda define outras notações, dependendo de quem inicia:

$\Gamma_g^A(G)$: número guloso de jogo quando Alice inicia.

$\Gamma_g^B(G)$: número guloso de jogo quando Bob inicia.

O artigo conclui limites superiores para o número guloso de jogo para duas classes de grafos:

- Para florestas, foi provado que $\Gamma_g(G) \leq 3$.
- Para grafos do tipo 2-árvore-parcial, foi provado que $\Gamma_g(G) \leq 7$.

4.2 PSPACE-completeness of two graph coloring games

Depois de quase trinta anos, em (COSTA *et al.*, 2020), é respondida uma questão em aberto proposta por Bodlaender: o número cromático de jogo (*game chromatic number*) é PSPACE-difícil. Ademais, o artigo também prova que o número guloso de jogo (*game Grundy number*) (HAVET; ZHU, 2013) também é PSPACE-difícil. É concluído que ambos os problemas são PSPACE-completos, mesmo que o número de cores seja igual ao número cromático.

O artigo também apresenta resultados sobre classes específicas de grafos, como os grafos *split* e várias superclasses de cografos (*cographs*). Foi provado para essas classes que o número guloso de jogo é igual ao número cromático ($\Gamma_g(G) = \chi_g(G)$), estendendo um dos resultados obtidos em (HAVET; ZHU, 2013).

Ao final, o artigo deixa a seguinte questão em aberto: $\Gamma_g(G) \leq 3$ é PSPACE-completo

para grafos gerais? Isto é, o Jogo da Coloração Gulosa é PSPACE-completo com 3 cores?

4.3 PSPACE-hardness of variants of the graph coloring game

Em (LIMA *et al.*, 2022), é provada a PSPACE-completude de cinco variantes do *Jogo da Coloração*, propostas por (ANDRES; LOCK, 2019 apud LIMA *et al.*, 2022, p. 1). Sendo essas variantes:

- $g_{A,A}$, na qual Alice inicia o jogo e pode pular turnos.
- g_B, na qual Bob inicia o jogo.
- $g_{B,B}$, na qual Bob inicia o jogo e pode pular turnos.
- $g_{A,B}$, na qual Alice inicia o jogo e Bob pode pular turnos.
- $g_{B,A}$, na qual Bob inicia o jogo e Alice pode pular turnos.

O artigo mostra que essas variantes são classificadas como problemas PSPACE-completos mesmo se o número de cores é igual ao número cromático.

O artigo também estuda duas variações do *Jogo da Coloração Conectada*, proposto em (CHARPENTIER *et al.*, 2020 apud LIMA *et al.*, 2022, p. 1). O jogo é semelhante ao *Jogo da Coloração*, com a condição de que o subgrafo induzido pelo conjunto de vértices coloridos deve ser conexo em todos os momentos do jogo. O artigo prova que os jogos são PSPACE-completos nas variantes que Alice ou Bob iniciam o jogo.

4.4 The connected greedy coloring game

Recentemente, em (LIMA *et al.*, 2023), as cinco variações do *Jogo da Coloração* que foram provadas serem problemas PSPACE-completos em (LIMA *et al.*, 2022), foram expandidas para o *Jogo da Coloração Gulosa* e também foram provadas serem problemas PSPACE-completos, mesmo se o número de cores for igual ao número cromático.

Além do mais, o artigo também introduz o *Jogo da Coloração Gulosa Conexa*, baseada em (CHARPENTIER *et al.*, 2020 apud LIMA *et al.*, 2023, p. 1). No jogo, os vértices devem ser coloridos com a menor cor possível (menor inteiro positivo) de um conjunto Cr de cores e o subgrafo induzido de vértices coloridos deve ser conexo, sendo as variações desse jogo:

- cgg_A, na qual Alice começa.
- cgg_B, na qual Bob começa.

Ambas as variações foram classificadas como problemas PSPACE-completos no artigo.

O artigo também mostra que, em grafos *split*, as duas variantes do *Jogo da Coloração Gulosa Conexa* são solucionáveis em tempo polinomial.

4.5 Conclusão do Capítulo

Os trabalhos revisados demonstram a complexidade do *Jogo da Coloração Gulosa*, o qual é o foco deste trabalho, e de jogos de coloração no geral. Muitos resultados interessantes acerca de complexidade foram apresentados, incluindo provas sobre a PSPACE-completude de algumas variações dos jogos. Este trabalho busca contribuir com essa linha de pesquisa ao analisarmos a PSPACE-completude do *Jogo da Coloração Gulosa* quando explorada com 3 cores, uma vez que os números de cores dos resultados de complexidade conhecidos são bem grandes, mesmo para instâncias pequenas.

5 METODOLOGIA

Neste capítulo, falaremos sobre a metodologia deste trabalho. Na Seção 5.1, definimos o PROBLEMA DO JOGO DA COLORAÇÃO GULOSA COM 3 CORES (PJCG3), na Seção 5.2 definimos o PROBLEMA POS-CNF, dando exemplo para um melhor entendimento, na Seção 5.3 mostramos uma tentativa de redução polinomial do POS-CNF para o PJCG3, na Seção 5.4 mostramos que $\Gamma_g(C_n) \leq 3$, na Seção 5.5 mostramos um problema na tentativa de redução da Seção 5.3, na Seção 5.6 mostramos uma tentativa de redução com o problema POS-CNF-6 e outra tentativa com grafos água-vivas na Seção 5.7. Por fim, mostramos nossos desenvolvimentos acerca do Algoritmo MiniMax.

5.1 O Problema do Jogo da Coloração Gulosa com 3 Cores

Nesta seção, definiremos o PROBLEMA DO JOGO DA COLORAÇÃO GULOSA COM 3 CORES conforme descrito abaixo:

Problema do Jogo da Coloração Gulosa com 3 cores

Instância: um grafo $G = (V, E)$.

Pergunta: Alice possui uma estratégia vencedora com 3 cores, seguindo as regras do jogo?

Note que o PROBLEMA DO JOGO DA COLORAÇÃO GULOSA COM 3 CORES (PJCG3) está em PSPACE, uma vez que, facilmente, podemos ver o Algoritmo 1 (adaptação do Algoritmo 1 de (COSTA, 2022, p. 47)), que verifica todas as possibilidades de jogadas e que usa apenas a memória do grafo e a memória das cores de cada vértice. Caso o grafo esteja sendo representado via matriz de adjacência, a memória terá tamanho $O(n^2)$. A memória das cores dos vértices pode ser um vetor de tamanho $n = O(n)$, com um resultado de espaço de $O(n^2)$. Esse algoritmo também pode ser visto como um algoritmo MiniMax para o PJCG3 e sua implementação na linguagem Python está no Apêndice B.

Agora, para concluirmos a PSPACE-completude, basta escolher um problema que seja conhecidamente PSPACE-completo e mostrarmos uma redução polinomial dele para o PJCG3. Para isso, escolhemos o PROBLEMA POS-CNF, definido a seguir.

Algoritmo 1: Resolve o PJCG3 em espaço polinomial

Entrada: Grafo G, jogada (Alice ou Bob).
Saída: responde SIM, caso exista uma estratégia vencedora para Alice em G com 3 cores e, em caso contrário, NÃO.

```
se G está totalmente colorido com 3 cores então
    retorne SIM
se vértice que não pode ser colorido com uma cor de Cr = {1,2,3} então
    retorne NÃO
se jogada = Alice então
    para cada vértice não colorido v ∈ V(G) faça
        para cada cor i = 1,2,3 faça
            se i não é cor de vértice vizinho de v então
                colorir v com cor i
                se Algoritmo 1 (G, Bob) = SIM então
                    retorne SIM
        descolorir v
    retorne NÃO
se jogada = Bob então
    para cada vértice não colorido v ∈ V(G) faça
        para cada cor i = 1,2,3 faça
            se i não é cor de vértice vizinho de v então
                colorir v com cor i
                se Algoritmo 1 (G, Alice) = NÃO então
                    retorne NÃO
        descolorir v
    retorne SIM
```

5.2 O Problema POS-CNF

Dado um conjunto $\{X_1, \ldots, X_N\}$ de N variáveis e uma fórmula na forma normal conjuntiva que possui M cláusulas $C_1, \ldots, C_M$, de forma que as cláusulas são formadas apenas por literais positivos, ou seja, não há a negação de nenhuma das variáveis, temos, dessa maneira, as instâncias do PROBLEMA POS-CNF, conforme descrito por Costa em sua tese (COSTA, 2022).

Assim como nos jogos de coloração, Alice e Bob também participam como jogadores no jogo referente ao PROBLEMA POS-CNF, alternando suas jogadas e atribuindo os valores verdadeiro ou falso aos literais que ainda não possuem valor. Alice ganha se, e somente se, a fórmula é verdadeira após todas as N variáveis terem sido valoradas em verdadeiro ou falso.

Como existem apenas literais positivos, Alice vai sempre valorar as variáveis escolhidas com Verdadeiro (V), enquanto Bob, que busca atrapalhar Alice, irá sempre atribuir Falso (F).

5.2.1 Exemplo do jogo referente ao PROBLEMA POS-CNF

Considerando a fórmula $\phi = (X_1 \vee X_2) \wedge (X_1 \vee X_3) \wedge (X_2 \vee X_4) \wedge (X_3 \vee X_4)$, nesta subseção, daremos alguns exemplos de jogadas.

- Caso Alice jogue X_1 **V**, Bob pode jogar X_4 **F**. Dessa forma, Bob vence, pois Alice não consegue satisfazer as cláusulas $(X_2 \vee X_4)$ e $(X_3 \vee X_4)$ numa única jogada;
- Caso Alice jogue X_2 **V**, Bob pode jogar X_3 **F**. Dessa forma, Bob vence, pois Alice não consegue satisfazer as cláusulas $(X_1 \vee X_3)$ e $(X_3 \vee X_4)$ numa única jogada;
- Caso Alice jogue X_3 **V**, Bob pode jogar X_2 **F**. Dessa forma, Bob vence, pois Alice não consegue satisfazer as cláusulas $(X_1 \vee X_2)$ e $(X_2 \vee X_4)$ numa única jogada;
- Caso Alice jogue X_4 **V**, Bob pode jogar X_1 **F**. Dessa forma, Bob vence, pois Alice não consegue satisfazer as cláusulas $(X_1 \vee X_2)$ e $(X_1 \vee X_3)$ numa única jogada;

Pelas jogadas anteriores, é possível notar que Bob possui uma estratégia vencedora, que consiste em fazer com que uma combinação de duas cláusulas fique vulnerável e sempre reste uma cláusula insatisfeita, fazendo, assim, com que Bob ganhe.

5.3 Tentativa de Redução Polinomial com POS-CNF

Nesta seção, descreveremos uma ideia inicial para a redução polinomial a partir de uma instância $< \phi >$ do PROBLEMA POS-CNF para uma instância $< G_\phi >$ do PROBLEMA DO JOGO DA COLORAÇÃO GULOSA COM 3 CORES. Para um melhor entendimento, tomaremos como exemplo a fórmula lógica com as quatro cláusulas C_1, C_2, C_3 e C_4 abaixo

$$\psi = \underbrace{(X_1 \vee X_2 \vee X_5)}_{C_1} \wedge \underbrace{(X_2 \vee X_3)}_{C_2} \wedge \underbrace{(X_3 \vee X_4)}_{C_3} \wedge \underbrace{(X_1 \vee X_4 \vee X_2)}_{C_4}$$

que é base para a criação do grafo exemplo G_ψ da Figura 9.

5.3.1 Construção do grafo G_ϕ da Redução Polinomial

(I) crie um vértice v (vértice verde de G_ψ)

(II) para cada cláusula $C = (y_1 \vee y_2 \vee \cdots \vee y_k)$ (chamamos essa criação de *ciclo cláusula* e os vértices criados aqui de *vértices cláusula*)

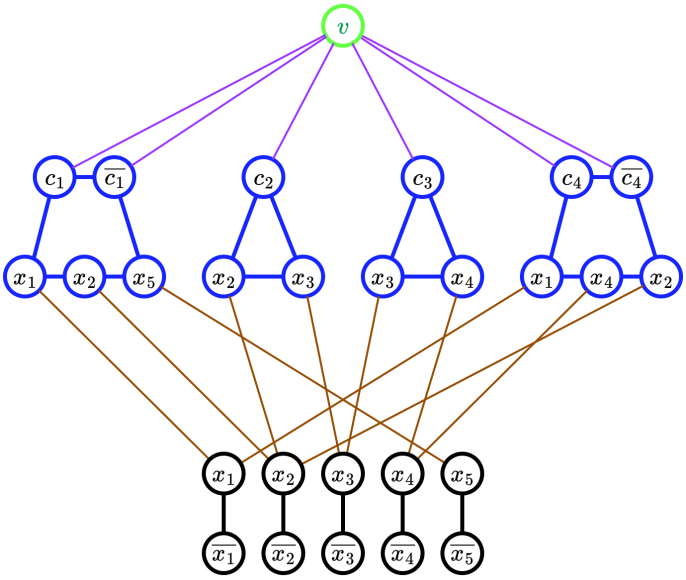
  - caso k seja par, criar o ciclo ímpar $(y_1, y_2, \cdots, y_k, c)$ (ciclos azuis de G_ψ criados a partir de C_2 e C_3)
  - caso k seja ímpar, criar o ciclo ímpar $(y_1, y_2, \cdots, y_k, \overline{c}, c)$ (ciclos azuis de G_ψ criados a partir de C_1 e C_4)

(III) para cada variável x, criar vértices x e $\overline{x}$ e ligá-los entre si (vértices e arestas pretas de G_ψ)

(IV) ligar o vértice v com os vértices c's e $\overline{c}$'s dos ciclos cláusula (arestas roxas de G_ψ)

(V) ligar os vértices variável aos vértices cláusula que correspondem ao mesmo literal (arestas de cor marrom de G_ψ)

Figura 9 – Grafo G_ψ exemplo da redução polinomial



Fonte: elaborado pelo autor via *draw.io* (JGRAPH, 2021).
Nota: aqui, as cores das linhas dos vértices são apenas para enfatizar a explicação e não representam a coloração do grafo.

5.3.1.1 Tempo para construção do Grafo G_ϕ

A construção do grafo pode ser feita em tempo polinomial. De fato, sendo

- v o número de variáveis (em nosso exemplo do grafo G_ϕ temos $v = 5$, pois temos as variáveis x_1, x_2, x_3, x_4 e x_5);
- c o número de cláusulas (em nosso exemplo do grafo G_ϕ temos $c = 4$, pois temos as cláusulas C_1, C_2, C_3 e C_4);
- ℓ o número de literais (em nosso exemplo do grafo G_ϕ temos $\ell = 10$, pois temos 3 literais em $C_1 = (X_1 \vee X_2 \vee X_5)$, 2 literais em $C_2 = (X_2 \vee X_3)$, 2 literais em $C_3 = (X_3 \vee X_4)$ e 3 literais em $C_4 = (X_1 \vee X_4 \vee X_2)$, que, ao somarmos, resultam num total de 10).

Agora, iremos focar na quantidade de vértices criados nos passos da construção.

- Em (II), criamos um *vértice cláusula* para cada literal da fórmula e, no máximo, mais dois *vértices cláusula* para cada cláusula, que serão o c e o $\overline{c}$, o que nos da $\ell + 2c = O(\ell + c)$ vértices criados em (II).
- Em (III), criamos 2 *vértices variável* para cada variável, o que nos dá $2v = O(v)$ vértices.

Dessa maneira, teremos o total de $O(\ell + c + v)$ vértices (um vértice de (I), mais $O(\ell + c)$ vértices de (II), mais $O(v)$ vértices de (III)).

Dando procedência, focaremos na quantidade de arestas criadas.

- Em (II), temos $\ell + 2c = O(\ell + c)$ arestas, aplicado o Lema de Handshaking (Wikipedia contributors, 2024) em todos os *ciclos cláusulas* juntos.
- Em (III), criamos uma aresta para cada uma das variáveis, o que nos dá $v = O(v)$ arestas.
- Em (IV), criamos, no máximo, duas arestas para cada ciclo, o que resultará em $2c = O(c)$ arestas.
- Em (V), criamos uma aresta para cada literal, o que nos dá $\ell = O(\ell)$ arestas.

Assim, teremos $O(\ell + c + v)$ arestas no total ($O(\ell + c)$ de (II), mais $O(v)$ de (III), mais $O(\ell)$ de (V)).

Portando, criamos um grafo com $O(\ell + c + v)$ vértices e $O(\ell + c + v)$ arestas, o que resulta num tempo de criação polinomial $O(\ell + c + v)$.

5.3.2 Estratégias padrões para Alice e Bob no grafo da construção G_ϕ

Uma estratégia vencedora para Bob, no grafo da construção, é colorir a vizinhança de algum *ciclo cláusula* com a cor 1, pois, deste modo, o ciclo deverá ser colorido obrigatoriamente com 3 cores diferentes da cor 1, porque todo *ciclo cláusula* é um ciclo ímpar, o que resultará em 4 cores no jogo, ou seja, Bob cumpre seu objetivo de forçar mais de 3 cores. Por exemplo, para o grafo G_ψ da Figura 9, se v (vértice verde) e os *vértices variáveis* pretos x_2 e x_3 estiverem com a cor 1, então o *ciclo cláusula* de C_2 terá que ser colorido com 3 cores distintas da cor 1. Com isso, podemos concluir que é vantajoso para Bob reproduzir a estratégia a seguir:

> *Estratégia padrão para Bob*: iniciar o jogo colorindo v (cor 1) e seguir colorindo os vértices variáveis x_i's.

Uma forma para que Alice impeça a estratégia padrão do Bob descrita acima é colorir os vértices variáveis $\overline{x_i}$ antes que seu vizinho seja colorido, pois isso garantirá que $\overline{x_i}$ tenha a cor 1 e, consequentemente, seu vizinho x_i terá cor diferente da cor 1, impedindo a estratégia de Bob.

A esta estratégia, nomeamos de *estratégia padrão de Alice*.

A relação do PJCG3 na redução G_ϕ com o PROBLEMA POS-CNF com instância $< \phi >$ é de que, quando Alice valorar um literal x_i em ϕ, consideramos que Alice colore $\overline{x_i}$ em G_ϕ. Já quando Bob valorar x_i em ϕ, consideramos que Bob colore o *vértice variável* x_i em G_ϕ.

Para concluir a PSPACE-completude, é necessário mostrar que Alice possui uma estratégia vencedora em $< \phi >$ se, e somente se, ela possui uma estratégia vencedora em G_ϕ no PROBLEMA DO JOGO DA COLORAÇÃO GULOSA COM 3 CORES na variação em que o Bob inicia o jogo. Embora isso seja intuitivo considerando os dois últimos parágrafos, infelizmente, a redução não funciona para alguns casos, como veremos adiante.

5.4 Jogo da Coloração Gulosa em Ciclos

O resultado mostrado nesta seção é simples, mas em nossos conhecimentos não o observamos na literatura. O resultado será necessário para um bom entendimento da seção seguinte.

Lema 5.4.1.

$$\Gamma_g(C_n) = \begin{cases} 3 & , \textit{se } n = 3 \\ 2 & , \textit{se } n = 4 \\ 3 & , \textit{se } n \geq 5 \end{cases}$$

Demonstração. Dado um C_3, em cada jogada, nota-se que uma nova cor será atribuída para cada vértice. Logo, $\Gamma_g(C_3) = 3$.

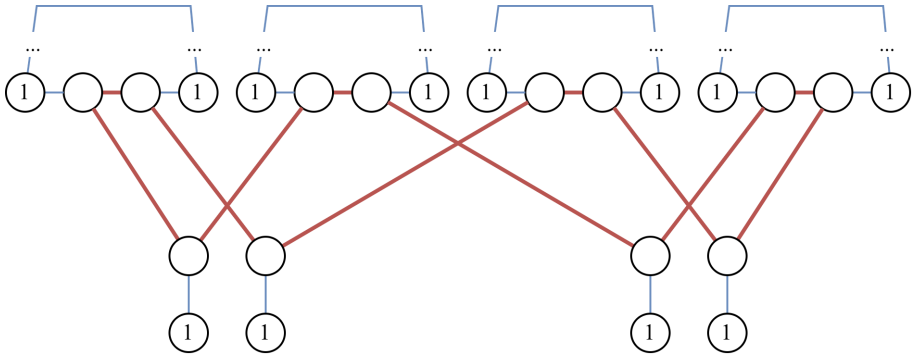
Dado um C_4, ao escolhermos o primeiro vértice, ele terá a cor 1 atribuída e, em qualquer outro momento do jogo, seus vizinhos terão a cor 2. Já o quarto vértice, que não é vizinho do primeiro colorido, terá a cor 1. Logo, $\Gamma_g(C_4) = 2$.

Dado um C_n com vértices $v_1, v_2, v_3, \cdots, v_n$ ligado na sequência descrita, onde $n \geq 5$, suponnha, sem perda de generalidade, que a primeira jogada seja $A(v_1)$ (cor 1), então, a jogada seguinte pode ser $B(v_4)$ (cor 1), o que implica v_2 e v_3 terem duas cores (2 e 3) diferentes da cor 1. A ideia principal de Bob é deixar as extremidades de um P_4 induzido com a cor 1 e fazer com que seus dois vértices internos tenham mais duas cores (como visto na Seção 3.6.1.1). Dessa maneira, Alice precisa de três cores para vencer, isto é, $\Gamma_g(C_n) = 3$ para $n \geq 5$. □

5.5 Problema na redução do POS-CNF

Na redução do POS-CNF, como as cláusulas podem ter qualquer tamanho, Bob pode forçar a configuração de jogo conforme a Figura 10, desde que a fórmula seja suficientemente grande. Para isso, basta que algumas cláusulas possuam mais que 7 literais, fazendo com que os *ciclos cláusula* tenham P_7 induzidos. Em vista disso, Bob pode escolher o vértice central do P_7 induzido e, em sua próxima jogada, selecionar um vértice das extremidades do P_7. Em decorrência disso, Bob obtém uma configuração de jogo com P_4 induzidos, cujas extremidades possuem a cor 1. Os demais vértices com cor 1 podem seguir das estratégias padrões de Bob e Alice apresentadas na Seção 5.3.2.

Figura 10 – Possível problema na redução do POS-CNF



Fonte: elaborado pelo autor via *draw.io* (JGRAPH, 2021).

Observa-se que os vértices não coloridos no grafo da Figura 10 induzem um ciclo C_{12}, destacado pelas arestas em vermelho, com toda a vizinhança colorida com a cor 1. Pelo Lema 5.4.1, conclui-se que Alice necessita de mais três cores diferentes da cor 1 para vencer, o que resulta em sua derrota.

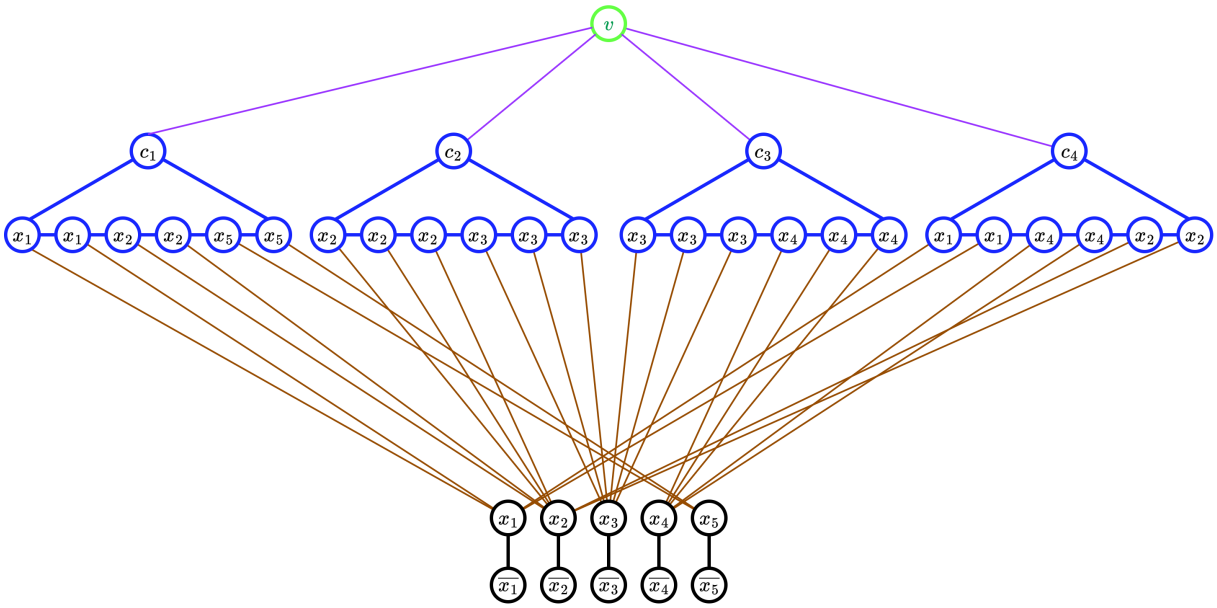
A partir dessa análise, investigamos outro problema e encontramos o PROBLEMA POS-CNF-6, que é semelhante ao PROBLEMA POS-CNF (Seção 5.2), mas com a restrição de que, em cada cláusula na fórmula, devem haver exatamente 6 literais. Em nossa pesquisa, esse é o tipo de problema PSPACE-completo com menos literais em cada cláusula conhecido.

5.6 Tentativa de redução com POS-CNF-6

O PROBLEMA POS-CNF-6 é PSPACE-completo (RAHMAN; WATSON, 2023) e, utilizando-se dele com a mesma redução do POS-CNF, não temos mais o empecilho obtido no PROBLEMA POS-CNF original, que permitia cláusulas de tamanho grande e possibilitava a

estratégia de Bob descrita anteriormente.

Figura 11 – Grafo G_ψ exemplo da redução do POS-CNF-6.



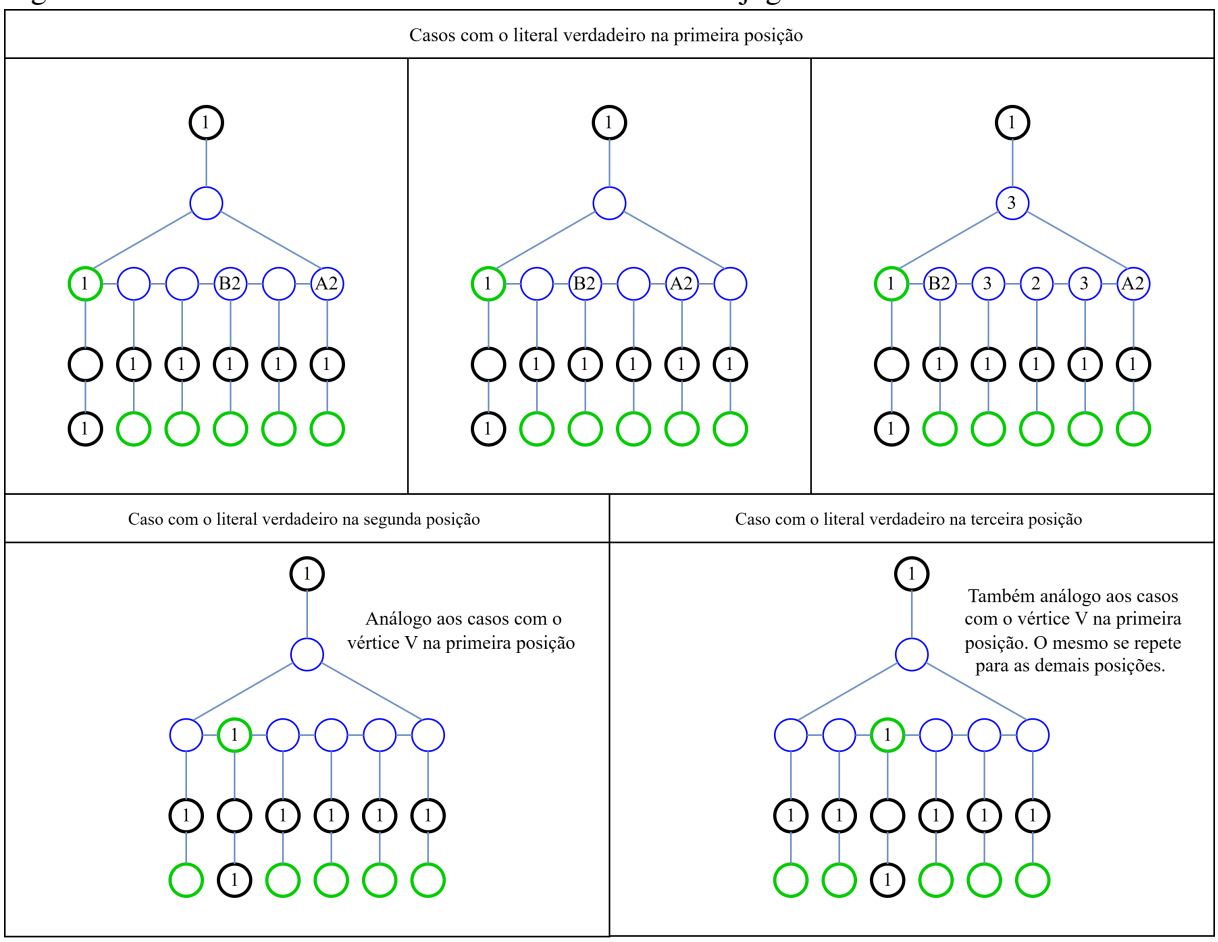
Fonte: elaborado pelo autor via *draw.io* (JGRAPH, 2021).
Nota: $\psi = (X_1 \vee X_1 \vee X_2 \vee X_2 \vee X_5 \vee X_5) \wedge (X_2 \vee X_2 \vee X_2 \vee X_3 \vee X_3 \vee X_3) \wedge (X_3 \vee X_3 \vee X_3 \vee X_4 \vee X_4 \vee X_4) \wedge (X_1 \vee X_1 \vee X_4 \vee X_4 \vee X_2 \vee X_2)$.

Considere os casos em que Alice vence no jogo POS-CNF-6 com somente um literal verdadeiro numa cláusula e que ela seguiu a estratégia padrão (veja a Figura 12, que cobre os casos no grafo da redução). Nesta configuração de jogo, nota-se que podem existir alguns vértices que possuem uma cor fixa. Ou seja, a partir desta configuração, independentemente das jogadas que Alice e Bob fizerem, os vértices fixos terão sempre a mesma cor associada a eles (veja os vértices verdes em cada caso na Figura 12). Com isso, chamamos de f o vértice fixo correspondente à posição do literal que Alice valorou como V. É evidente que, nesta configuração de jogo, Alice também vencerá no *Jogo da Coloração Gulosa* nesta cláusula, seguindo a seguinte estratégia:

- se $B(v)$, onde $v \notin N(f)$, então $A(u)$, em que $u \notin N(v)$ tem distância dois de f.
- se $B(v)$, onde $v \in N(f)$, então $A(u)$, em que $u \neq f$ com distância dois de f.

Analisando os casos em que Alice vence no jogo POS-CNF-6 com dois literais verdadeiros numa única cláusula, onde os dois literais V não são vizinhos em relação à posição na cláusula, facilmente vemos que Alice segue vencendo. Veja a Figura 13.

Figura 12 – Casos com somente um literal verdadeiro no jogo POS-CNF-6



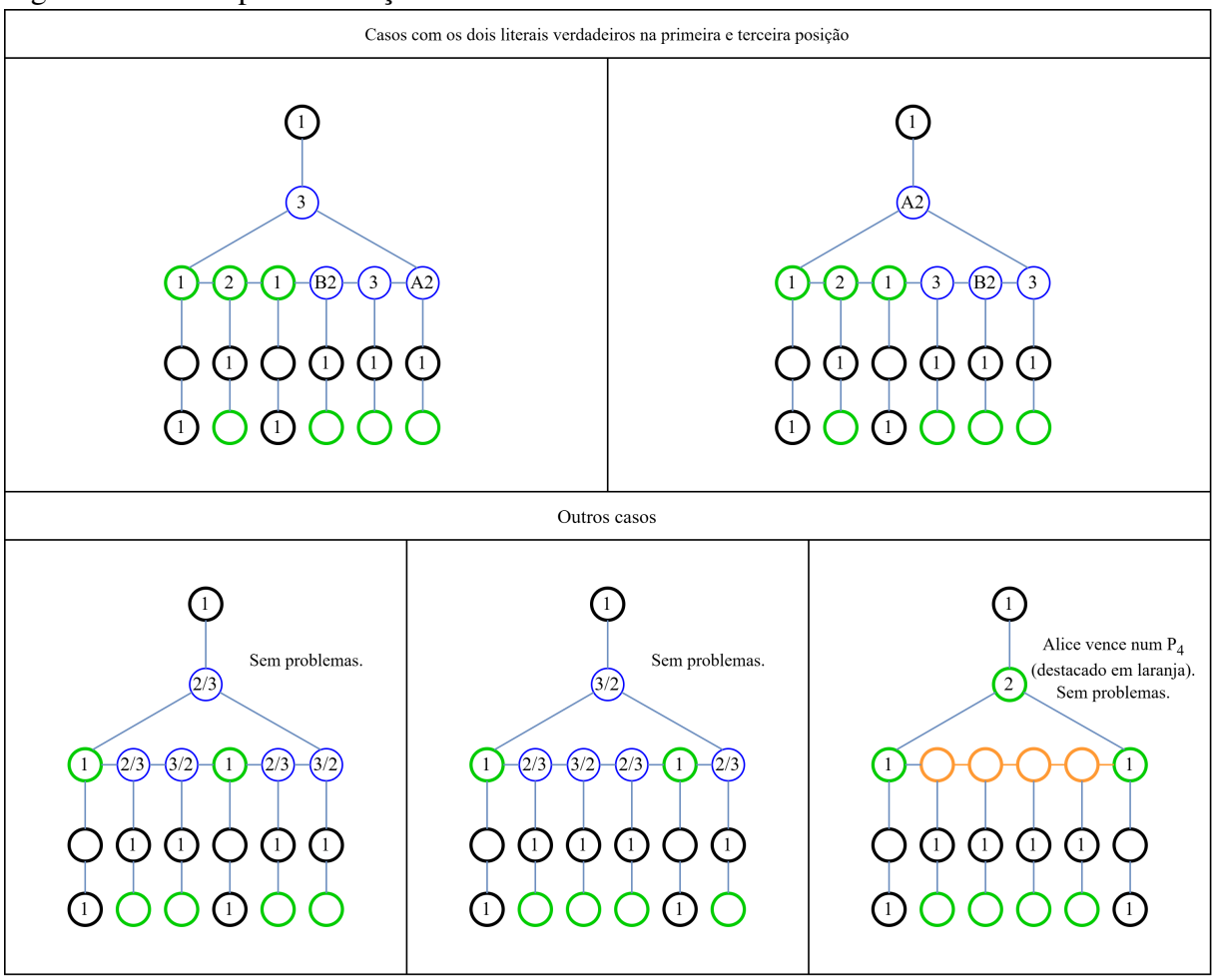
Fonte: elaborado pelo autor via *draw.io* (JGRAPH, 2021).
Nota: as cores das linhas dos vértices são apenas para ilustração; a coloração real está representada por valores inteiros. Os vértices fixos estão representados pelos vértices de linha verde. Ademais, os vértices que contém 'A' ou 'B' indicam que foram coloridos por Alice ou Bob, respectivamente.

5.7 Dificuldade na redução do POS-CNF-6

Contudo, ao analisar os casos simples onde Alice escolhe dois literais como verdadeiros e esses literais são vizinhos numa mesma cláusula (veja um exemplo dessa configuração na Figura 14), temos um problema no *Jogo da Coloração Gulosa* caso os *vértices variável* correspondentes aos dois literais escolhidos estejam ligados com vértices de cor 2 presentes em outros *ciclos cláusula* (representados pelos dois vértices menores de cor azul na Figura 14) do grafo, pois essa configuração se mostrou muito favorável para Bob, enquanto Alice, que deveria vencer nesse caso, não consegue uma estratégia vencedora. Acreditamos que essa configuração seja possível e não conseguimos abstrair uma estratégia para Alice contorná-la. Talvez a configuração seja inexistente a depender de alguma estratégia mais refinada da Alice.

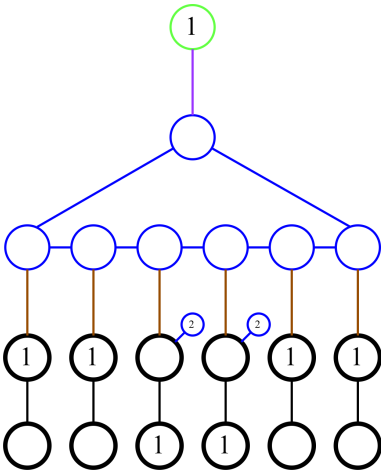
Observando a Figura 15, na qual Bob começa colorindo um vértice do *ciclo cláusula* com a cor 2, vemos que, logo após a jogada de Alice, Bob consegue uma estratégia vencedora ao

Figura 13 – Exemplo de redução do 6-POS-CNF



Fonte: elaborado pelo autor via *draw.io* (JGRAPH, 2021).
Nota: as cores entre barra (‘/’) indicam as possíveis opções de coloração para um vértice. Já os casos restantes são *espelhados* dos casos apresentados e, por isso, não estão na figura.

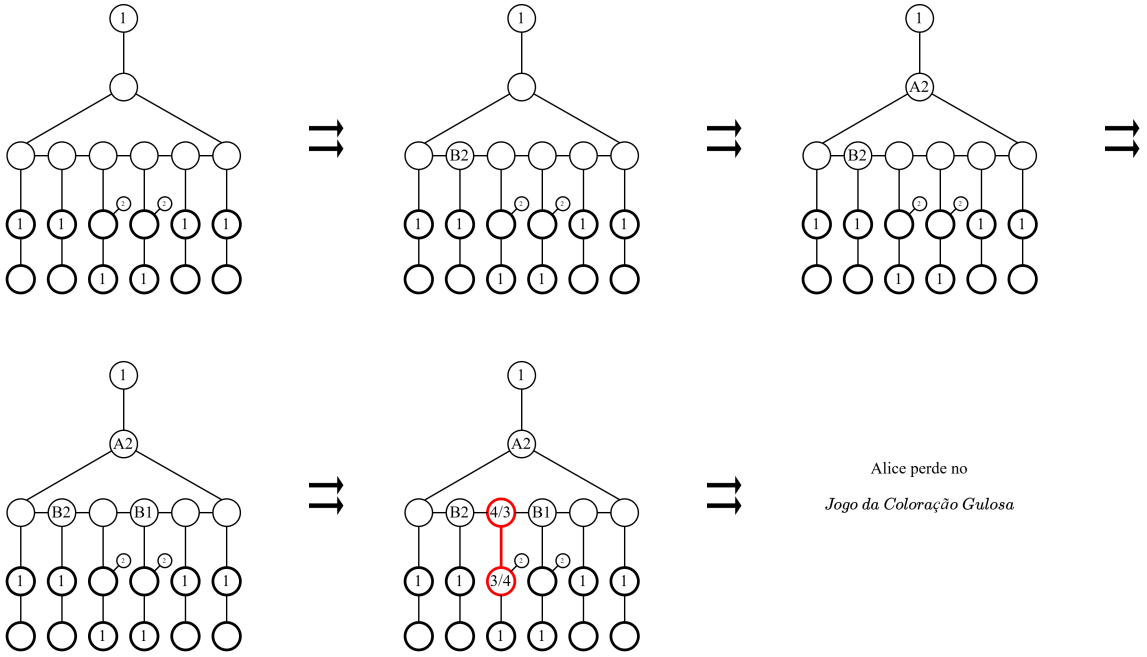
Figura 14 – Possível problema na redução do POS-CNF-6



Fonte: elaborado pelo autor via *draw.io* (JGRAPH, 2021).

"trancar" os vértices destacados pela linha vermelha, forçando-os a terem as cores 3 e 4, fugindo do nosso objetivo de utilizar, no máximo, 3 cores.

Figura 15 – Caso 1 de jogadas que levam a derrota de Alice no grafo de redução do POS-CNF-6



Fonte: elaborado pelo autor via *draw.io* (JGRAPH, 2021).
Nota: os vértices destacados pelas linhas vermelhas indicam o uso de uma quarta cor.

Para evitar a situação anterior, na Figura 16, temos os vértices que Alice pode jogar. Observe que, para fugir desse caso, Alice pode escolher somente um único vértice, que está destacado pela cor azul. Em seguida, vamos analisar mais um caso, onde Alice escolhe esse vértice após a jogada de Bob.

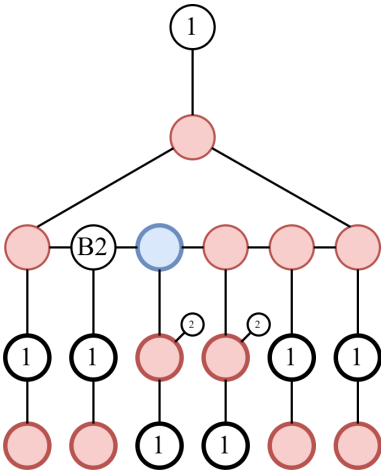
Agora, averiguando a Figura 17, onde Alice escolhe o vértice sugerido anteriormente, constatamos que Bob consegue, novamente, uma nova estratégia, forçando outros vértices a terem uma quarta cor.

Parecidamente com o Caso 1, na Figura 18(c), temos os vértices que Alice deve evitar jogar para não formar a configuração do caso 2. É possível notar que, de novo, Alice só pode escolher um único vértice para escapar da estratégia do caso 2.

Unindo os vértices que não são favoráveis para Alice no Caso 1 (Figura 16) e Caso 2 (Figura 18(c)), nota-se que, pela Figura 18(d), Alice não pode escolher nenhum vértice para colorir sem que Bob consiga uma estratégia vencedora.

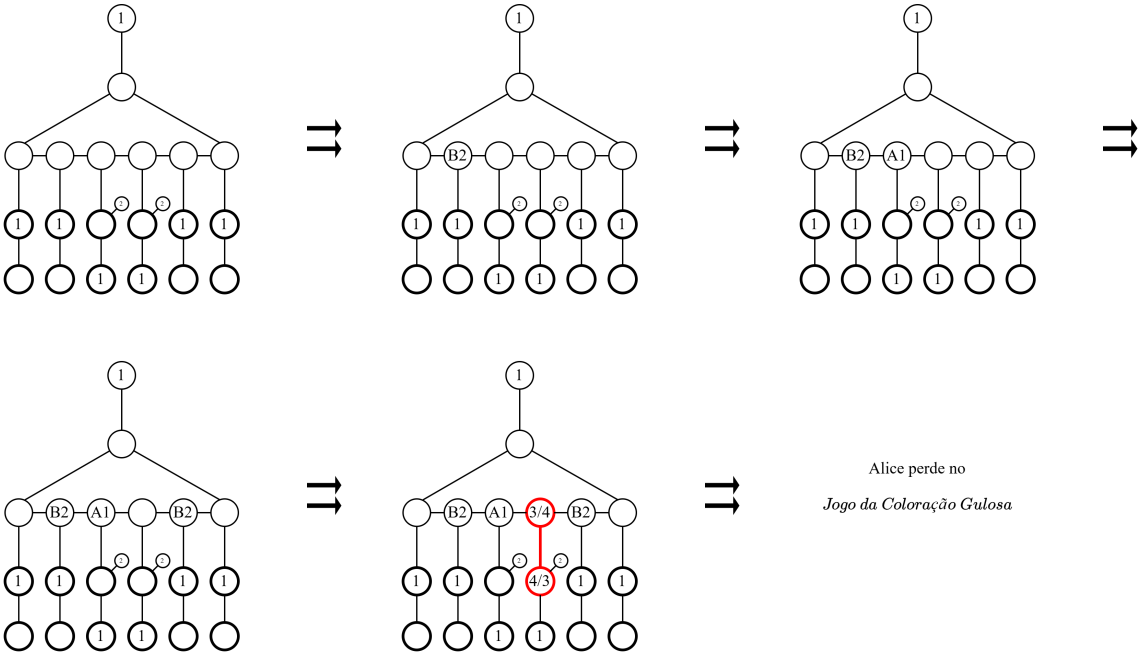
No entanto, ainda pode existir a indagação do que aconteceria caso Alice iniciasse nessa configuração, em vez de Bob. Uma jogada que ela poderia fazer de início seria impedir o

Figura 16 – Grafo com os vértices que Alice pode ou não escolher para evitar o Caso 1 da Figura 15



Fonte: elaborado pelo autor via *draw.io* (JGRAPH, 2021).
Nota: os vértices azuis indicam são favoráveis para Alice, enquanto os vermelhos não são.

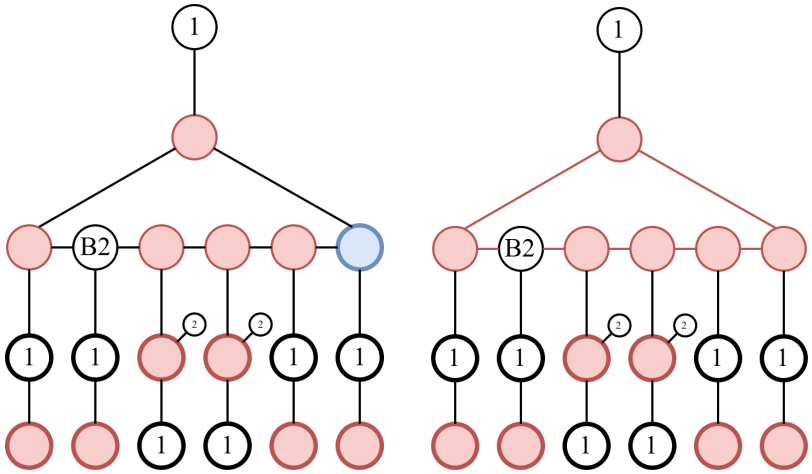
Figura 17 – Caso 2 de jogadas que levam a derrota de Alice no grafo de redução do POS-CNF-6



Fonte: elaborado pelo autor via *draw.io* (JGRAPH, 2021).
Nota: os vértices destacados pelas linhas vermelhas indicam o uso de uma quarta cor.

vértice com cor 2, no qual Bob iniciou nos casos anteriores, de existir. Veja na Figura 19 que, ao Alice escolher o primeiro *vértice cláusula* da parte inferior, Bob se aproveita do espelhamento do grafo para jogar no lado oposto, fazendo não só com que sua configuração favorável seja feita, como também garantindo sua vitória de imediato ao obrigar mais de 3 cores nos vértices

Figura 18 – Grafos com os vértices que Alice pode ou não escolher

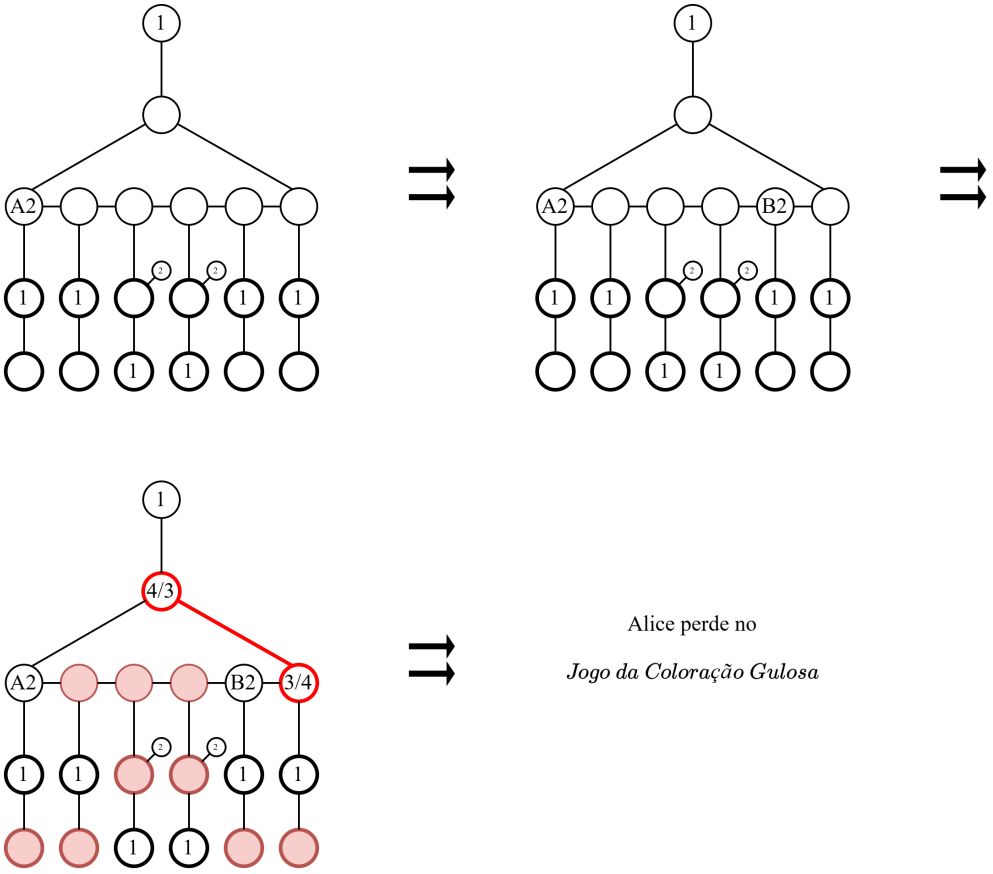


(c) Caso 2 da Figura 17 (d) Casos 1 e 2

Fonte: elaborado pelo autor via *draw.io* (JGRAPH, 2021).
Nota: o vértice azul é favorável para Alice, enquanto os vermelhos não são.

destacados.

Figura 19 – Jogadas quando Alice inicia na configuração problemática do grafo de redução do POS-CNF-6



Fonte: elaborado pelo autor via *draw.io* (JGRAPH, 2021).
Nota: os vértices destacados pelas linhas vermelhas indicam o uso de uma quarta cor. Enquanto que os vértices preenchidos em vermelho não são favoráveis para serem escolhidos por Alice.

Após essa análise, repara-se que, de fato, essa configuração de jogo é bastante favorável para o Bob. Mesmo que Alice ganhe pela fórmula, diante dessa situação, mostrou-se muito difícil obtermos uma abstração de jogo para saber como Alice poderia contornar os casos analisados previamente. Isto posto, tentamos com uma redução mais simples, com grafos *água-viva* (nomeados assim neste trabalho).

5.8 Tentativa de redução com grafo água-viva

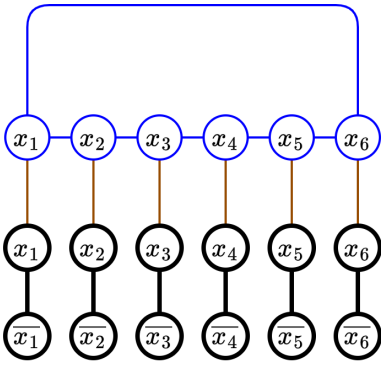
Um grafo água-viva é um grafo que possui um C_6 como subgrafo induzido e subgrafos P_3 com extremidades nos vértices do C_6, como na Figura 20. A redução polinomial (veja na Figura 20) a partir do POS-CNF-6 se dá por:

(I) para cada cláusula $C = (y_1 \vee y_2 \vee \cdots \vee y_6)$, criar ciclo $(y_1, y_2, \cdots, y_6)$ vértices (cláusulas) e arestas de cor azul.

(II) para cada variável x, criar vértices (variáveis) x e $\overline{x}$ e ligá-los entre si (vértices e arestas de cor preta).

(III) ligar os vértices variável aos vértices cláusula que correspondem ao mesmo literal (arestas de cor marrom).

Figura 20 – Grafo água-viva



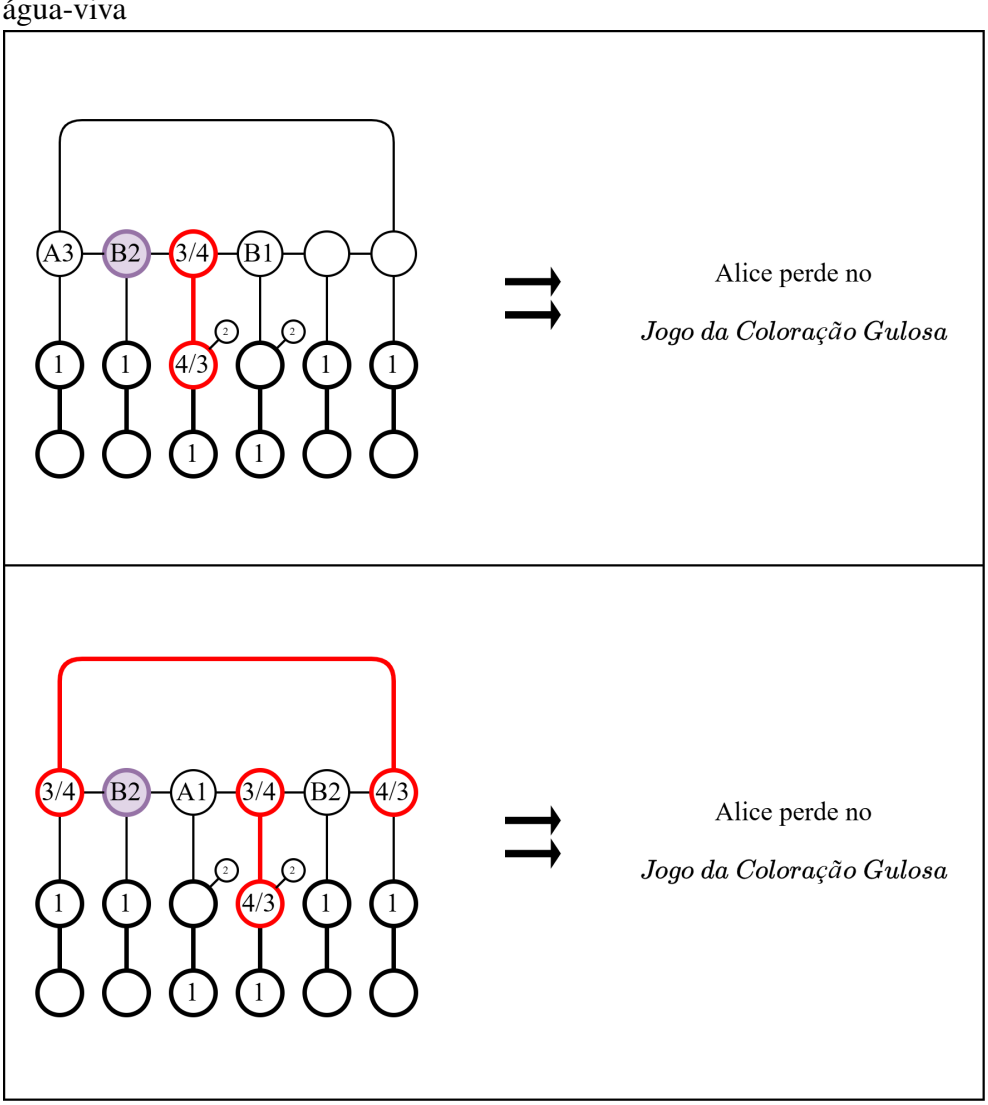
Fonte: elaborado pelo autor via *draw.io* (JGRAPH, 2021).
Nota: redução feita a partir da fórmula $\psi = (X_1 \vee X_2 \vee X_3 \vee X_4 \vee X_5 \vee X_6)$

5.9 Dificuldade na redução com grafo água-viva

Contudo, identificamos outra ocorrência de configuração problemática semelhante à do grafo da redução do POS-CNF-6, com dois literais verdadeiros vizinhos na cláusula. Nas

Figuras 21 (Bob inicia) e 22 (Alice inicia), vemos que essa é, também, uma configuração muito favorável para Bob. Pela Figura 23, observamos, mais uma vez, um ponto de estagnação para Alice.

Figura 21 – Casos onde Bob inicia na configuração problemática do grafo água-viva



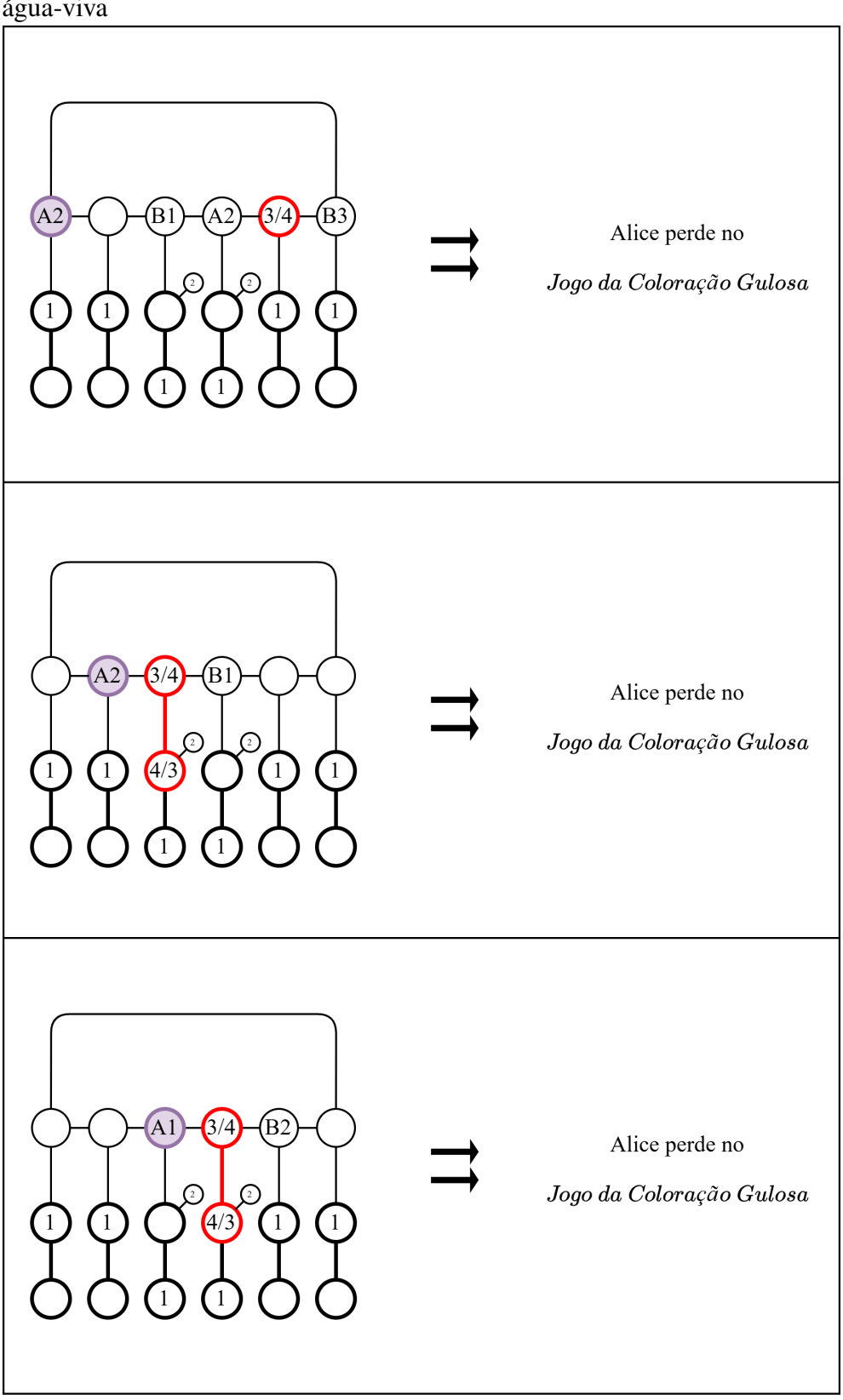
Fonte: elaborado pelo autor via *draw.io* (JGRAPH, 2021).
Nota: o vértice preenchido na cor roxa indica o primeiro vértice escolhido por Bob na configuração. Já os vértices destacados pelas linhas vermelhas indicam o uso de uma quarta cor.

Nesta situação, nossa abstração para configurações maiores ficou bastante difícil, então recorremos ao o uso do algoritmo MiniMax.

5.10 Uso do MiniMax

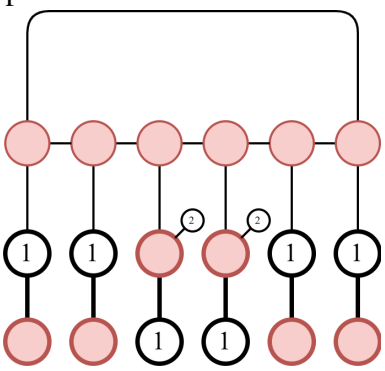
Com o intuito de facilitar a pesquisa ao analisarmos configurações de jogo mais complexas, implementamos, na linguagem de programação Python, o algoritmo MiniMax,

Figura 22 – Casos onde Alice inicia na configuração problemática do grafo água-viva



Fonte: elaborado pelo autor via *draw.io* (JGRAPH, 2021).
Nota: o vértice preenchido na cor roxa indica o primeiro vértice escolhido por Alice na configuração. Já os vértices destacados pelas linhas vermelhas indicam o uso de uma quarta cor. Os demais casos não estão presentes na figura porque são espelhados.

Figura 23 – Grafo água-viva com os vértices não favoráveis para Alice



Fonte: elaborado pelo autor via *draw.io* (JGRAPH, 2021).
Nota: os vértices preenchidos em vermelho não são favoráveis para serem escolhidos por Alice.

adaptando-o para o *Jogo da Coloração Gulosa*.

Todavia, antes de podermos utilizar o algoritmo MiniMax, foi necessário, primeiro, implementarmos a redução para o grafo. Para isso, seguimos a ideia da Seção 5.3.1.

5.10.1 Função de redução

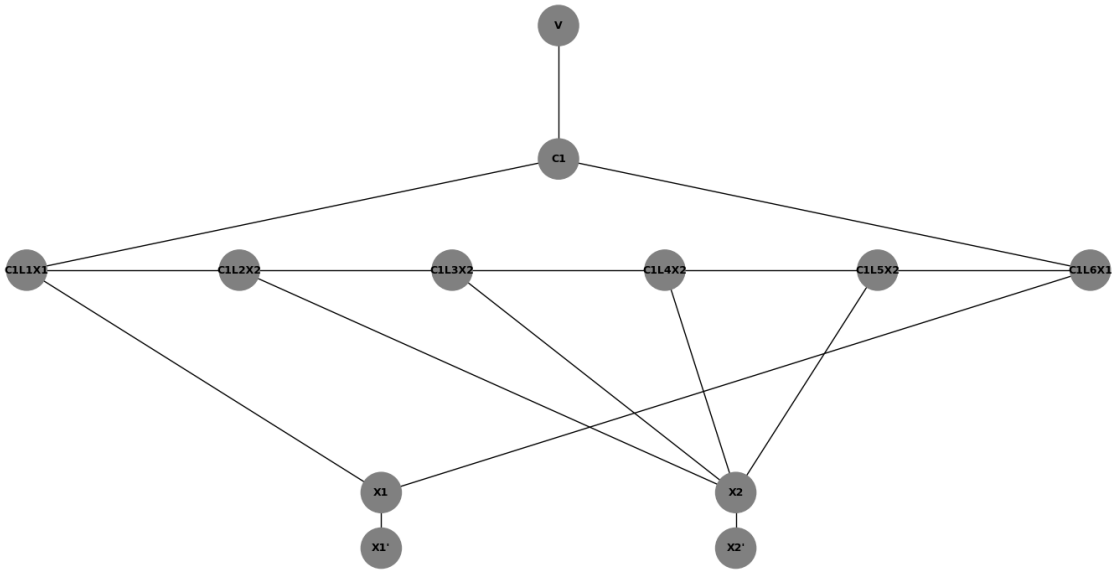
Para representarmos a fórmula do POS-CNF-6, utilizamos a estrutura *dicionário* do Python (Código-fonte 2), a qual é uma estrutura de dados que armazena pares chave-valor e permite o acesso eficiente aos dados através da chave (para mais detalhes, veja (Python Software Foundation, 2024)). De início, é indicada a quantidade de variáveis n para o algoritmo criar automaticamente as variáveis $X_1, X_2, X_3, \cdots, X_n$ (Linhas 1 e 11). Na Linha 12, é armazenada a valoração de cada variável. O valor *None* atribuído indica um valor nulo (ou vazio), pois, inicialmente, as variáveis não têm valoração.

A função *reducao* (Código-fonte 1) recebe uma fórmula na forma POS-CNF-6 e retorna um grafo construído a partir da redução. Para representar um grafo, utilizamos a *NetworkX*, a qual é uma biblioteca do Python para a criação, manipulação e estudo de estruturas de redes complexas (HAGBERG *et al.*, 2008).

Inicialmente, criamos o vértice v (Linha 3). Prosseguindo, criamos os vértices referentes as cláusulas (Linha 6) e, ainda na mesma iteração, na Linha 8, criamos o *ciclo cláusula* para a cláusula atual e, para facilitar a implementação, utilizamos outra nomenclatura para os *vértices cláusula* no lugar de $(y_1, y_2, \cdots, y_k)$. Depois, são criados os *vértices variável*, onde o

complemento das variáveis é representado por Xi' em vez de $\overline{x_i}$ (Linha 10). Finalizando, na Linha 11, definimos que cada vértice de G terá o atributo *color* (cor), fundamental para o nosso estudo, e que os vértices começarão com a cor cinza (*gray*), representando os vértices não coloridos (veja um exemplo de resultado na Figura 24).

Figura 24 – Grafo resultante da função *reducao* (Código-fonte 1)



Fonte: elaborado pelo autor via *Matplotlib* (HUNTER, 2007).
Nota: $\psi = (X_1 \vee X_2 \vee X_2 \vee X_2 \vee X_2 \vee X_1)$

Isto posto, podemos agora analisar de forma mais detalhada a função *minimax*.

5.10.2 Função minimax

Observando a função *minimax* (Código-fonte 5), percebemos que ela é baseada no Algoritmo 1, com a diferença de que quando Alice ou Bob encontram um caso favorável, eles *descolorem* o vértice antes de retornar na recursão (Linha 11 e Linha 19). Essa mudança é essencial para evitar a coloração inadequada de G que poderia acontecer em outras chamadas, o que afetaria a análise dos vértices não coloridos no nível atual da recursão.

Também é possível notar que a função tem outros retornos com a resposta de jogo que pode ser Verdadeiro (*True*) ou Falso (*False*). Esses demais retornos serão explicados quando falarmos sobre a implementação do *Jogo da Coloração Gulosa*.

Como caso base da recursão da função *minimax*, temos:

- Linha 2, a qual indica se G está totalmente colorido com as k cores (Código-fonte 8).
- Linha 4, a qual indica se existe algum vértice que não possa ser colorido com as k cores

(Código-fonte 9).

Na Linha 6, utilizando o (Código-fonte 10), selecionamos os vértices não coloridos de G e os analisamos a depender do jogador da vez.

Averiguando a vez de Alice jogar, iremos, primeiro, colorir o vértice não colorido atual (Linha 9). Logo depois, é analisado o resultado do MiniMax para quando Bob joga após a jogada atual de Alice. Caso a função *minimax* retorne Verdadeiro (*True*) (Linha 10), significa que o vértice não colorido atual é favorável para Alice. Logo, ela imediatamente retorna. Caso o retorno fosse Falso (*False*), o vértice atual seria descolorido (Linha 11) e a análise prosseguiria para o próximo vértice não colorido.

Se nenhum dos vértices não coloridos for favorável para Alice, significa que Alice perde na configuração de jogo para o nível atual de recursão. Portanto, na Linha 14, retornamos Falso. A análise para quando for a vez de Bob é análoga a da vez de Alice.

Agora que os códigos relacionados à redução da fórmula e à implementação do MiniMax foram discutidos, será apresentada a implementação de uma versão jogável do *Jogo da Coloração Gulosa*.

5.10.3 Implementação do Jogo da Coloração Gulosa

Na função *jogo_coloracao_gulosa* (Código-fonte 15), o usuário pode escolher com quem jogar (Alice ou Bob) (Linha 2). A varável de código *jogo* indica o resultado do jogo e recebe três possíveis valores:

0 , indicando que o jogo não terminou.

1 , caso Alice vença o jogo.

-1 , caso Bob vença o jogo.

Caso o usuário escolha Alice para jogar, ele deverá escolher um vértice não colorido presente em G para fazer sua jogada. Depois, a jogada da vez é transferida para Bob (Linha 27), que faz sua melhor jogada possível após a análise do MiniMax.

Uma cópia do grafo G é feita na Linha 37 para que não haja alterações nele durante a função *minimax* (Código-fonte 5). Na Linha 38, é feita a chamada do MiniMax para observar se Bob consegue vencer na configuração de jogo atual. Caso o retorno do MiniMax seja Verdadeiro, significa que Bob perde, então ele faz uma jogada em um vértice não colorido qualquer de G para dar procedência ao jogo (Linha 39).

No entanto, caso Bob vença após a jogada feita por Alice, ele faz sua jogada baseada

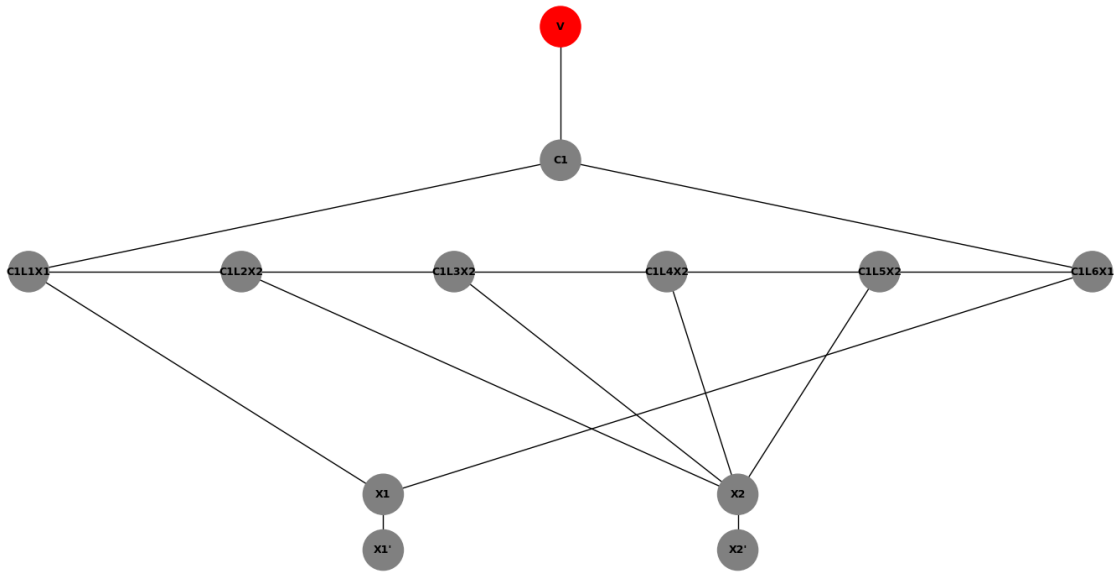
no vértice favorável a ele retornado pelo MiniMax (valor na variável de código *res[2]*), pois a função *minimax*, além de retornar o resultado (Verdadeiro ou Falso), pode retornar um vértice e cor favoráveis para Alice ou Bob (Linha 44).

A análise feita anteriormente é análoga para quando o usuário escolhe Bob como jogador.

Como exemplo de funcionamento do código, temos um caso onde o usuário escolhe jogar com Alice. Sendo as jogadas:

– Turno 01: Bob concluiu que perde, então colore V com vermelho (Figura 25).
– Turno 02: o usuário escolhe o vértice $X1'$ como Alice (Figura 26).
⋮
– Turno final: Bob encontrou uma estratégia vencedora fazendo com que o vértice $C1L3X2$ não possa ser colorido com 3 cores (Figura 27).

Figura 25 – Grafo resultante da função *jogo_coloracao_gulosa* (Código-fonte 15) no turno 1 de jogo

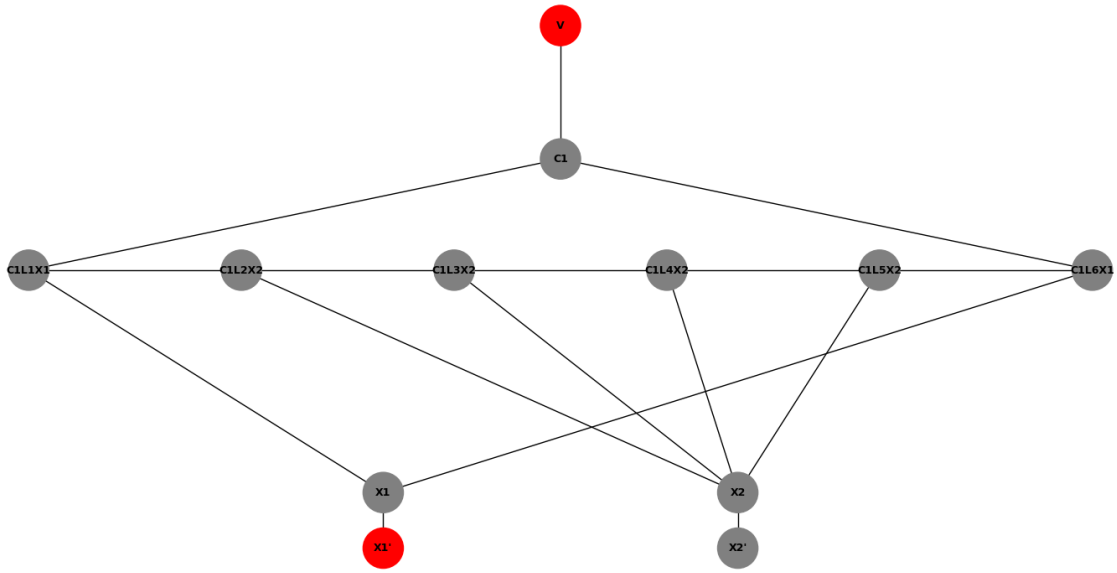


Fonte: elaborado pelo autor via *Matplotlib* (HUNTER, 2007).
Nota: $\psi = (X_1 \vee X_2 \vee X_2 \vee X_2 \vee X_2 \vee X_1)$

Como exemplo de grafo onde Alice consegue uma estratégia para vencer, temos a Figura 28.

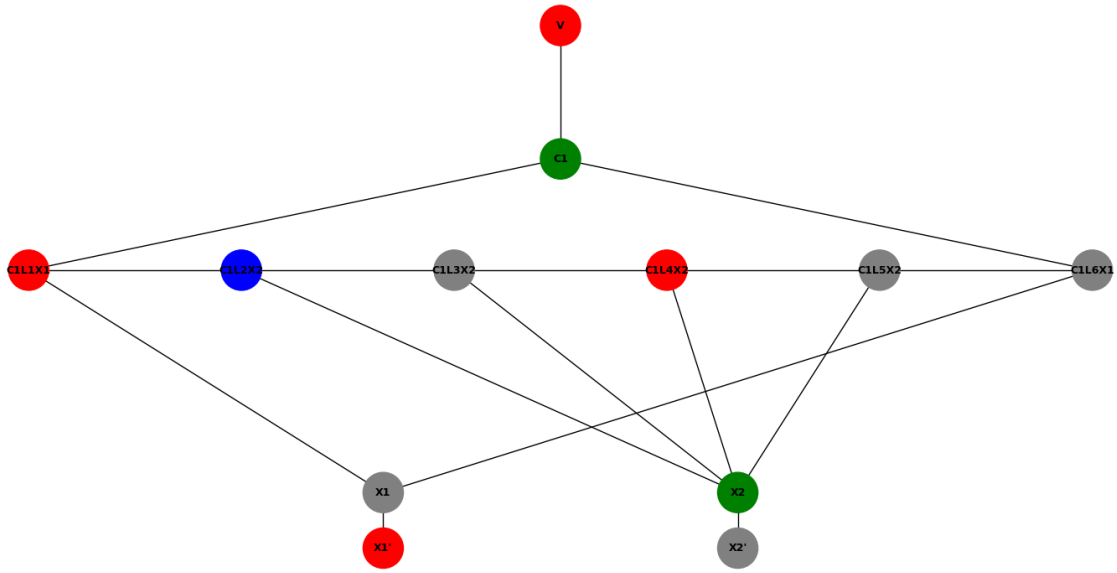
No entanto, por questões de tempo, não exploramos o uso da implementação do algoritmo MiniMax. Como previsto pela teoria, a implementação consome pouca memória (possui espaço polinomial), mas necessita de um tempo considerável (tempo exponencial) para a criação de uma árvore MiniMax para grafos criados a partir de fórmulas com apenas duas

Figura 26 – Grafo resultante da função *jogo_coloracao_gulosa* (Código-fonte 15) no turno 2 de jogo



Fonte: elaborado pelo autor via *Matplotlib* (HUNTER, 2007).
Nota: $\psi = (X_1 \vee X_2 \vee X_2 \vee X_2 \vee X_2 \vee X_1)$

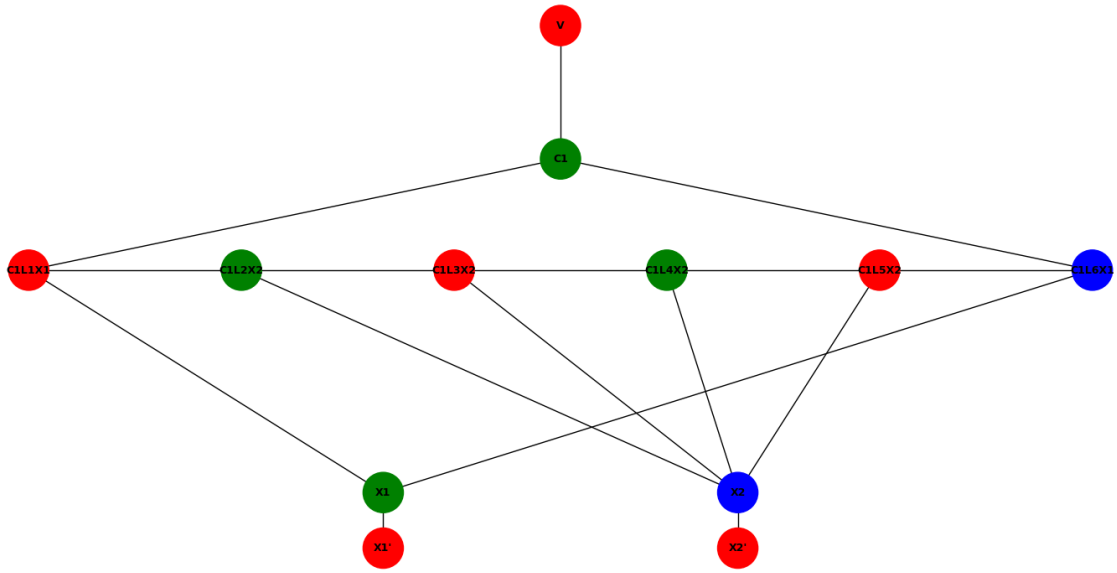
Figura 27 – Grafo resultante da função *jogo_coloracao_gulosa* (Código-fonte 15) no turno final de jogo



Fonte: elaborado pelo autor via *Matplotlib* (HUNTER, 2007).
Nota: $\psi = (X_1 \vee X_2 \vee X_2 \vee X_2 \vee X_2 \vee X_1)$

cláusulas. Para uma análise mais profunda, necessitamos de tempo para gerar fórmulas maiores e verificar se o MiniMax possui o mesmo retorno para ambos os jogos da redução.

Figura 28 – Grafo resultante da função *jogo_coloracao_gulosa* (Código-fonte 15) quando Alice vence o jogo



Fonte: elaborado pelo autor via *Matplotlib* (HUNTER, 2007).
Nota: $\psi = (X_1 \vee X_2 \vee X_2 \vee X_2 \vee X_2 \vee X_1)$

6 RESULTADOS

Neste capítulo, apresentamos os principais resultados obtidos durante a investigação da PSPACE-completude do Jogo da Coloração Gulosa com 3 cores.

6.1 Contribuições Parciais

Nesta seção, apresentamos nossas contribuições para responder à pergunta sobre o Objetivo Geral deste trabalho.

6.1.1 Análise de Reduções Polinomiais

Identificamos desafios em reduções de problemas PSPACE-completos (POS-CNF e POS-CNF-6) para grafos:

- **Cláusulas longas:** na redução do POS-CNF, identificamos uma estratégia vencedora para Bob que envolve cláusulas que têm mais de 7 literais. Essa observação nos levou à conclusão de que a redução do POS-CNF é inviável para demonstrar a PSPACE-completude com 3 cores (Seção 5.5).
- **Literais vizinhos:** nas reduções do problema POS-CNF-6 e do grafo água-viva, identificamos possíveis configurações problemáticas para quando Alice valora dois literais vizinhos com V. Por não sabermos provar matematicamente como Bob forçaria tais configurações, deixamos em aberto ambas as reduções (Seções 5.7 e 5.9).

A partir dessas identificações, acreditamos que elas servem como orientações para pesquisas futuras.

6.1.2 Lema do número guloso em ciclos 5.4.1

Como mais uma contribuição do nosso trabalho, temos o Lema 5.4.1, onde demonstramos que, para ciclos com $n \geq 5$ vértices, $\Gamma_g(C_n) = 3$. Acreditamos que esse lema possa ser útil para pesquisas com grafos *cactus*, os quais já sabemos que $\chi_g(G) = 5$ (SIDOROWICZ, 2007) para o *Jogo da Coloração* normal.

6.1.3 Ferramentas Computacionais

Por fim, temos uma descrição de como nossas implementações podem ajudar em outras pesquisas:

- **MiniMax:** a implementação do Algoritmo MiniMax (Seção 5.10.2) será de grande ajuda em novas tentativas de redução de problemas de fórmulas lógicas para problemas de jogos de coloração em grafos. A implementação também é de fácil atualização para outros jogos de coloração em grafos com tipos diferentes de coloração (coloração harmônica, por exemplo).
- ***Jogo da Coloração Gulosa*:** a implementação do jogo (do tipo *humano vs máquina*) também pode auxiliar na verificação em pesquisas ainda não comprovadas matematicamente, especialmente para grafos que são razoavelmente pequenos para máquinas, porém grandes aos olhos humanos.

Embora a questão central sobre a PSPACE-completude com 3 cores permaneça em aberto, consideramos que nossas descobertas e contribuições representam um avanço significativo na pesquisa.

7 CONCLUSÕES E TRABALHOS FUTUROS

A PSPACE-completude do PROBLEMA DO JOGO DA COLORAÇÃO GULOSA COM 3 CORES revelou-se bastante desafiadora, pois, mesmo em casos pequenos, tivemos dificuldade para abstrair as configurações de jogo. Apesar disso, este trabalho avançou na investigação da PSPACE-completude do *Jogo da Coloração Gulosa* com 3 cores, embora a questão central permaneça em aberto.

Conforme discutido nos Resultados (Capítulo 6), identificamos que certas estruturas, como cláusulas longas e literais vizinhos, são críticas para a análise das reduções. Além disso, o Lema 5.4.1 estabelece um limite teórico relevante quanto ao número guloso de jogo para ciclos. Já com relação às implementações do MiniMax e do jogo *humano vs máquina*, essas provaram-se muito úteis para este e para possíveis outros trabalhos, apesar das dificuldades causadas pelo tempo exponencial do MiniMax.

Para trabalhos futuros, deixamos o estudo das tentativas de redução aqui mostradas para outras variações do jogo (onde a Alice inicia, por exemplo). Deixamos também uma implementação do jogo *humano vs máquina* mais interativa, sem a necessidade de usar o terminal para efetuar jogadas, além de uma melhor otimização de tempo do MiniMax, em troca de um consumo maior de memória. Por fim, esperamos, em um futuro próximo, que seja alcançada a demonstração formal da PSPACE-COMPLETUDE DO JOGO DA COLORAÇÃO GULOSA COM 3 CORES, um problema que permanece em aberto, com o auxílio do POS-CNF-6 ou do grafo água-viva.

REFERÊNCIAS

ANDRES, D.; LOCK, E. Characterising and recognising game-perfect graphs. **Discrete Mathematics & Theoretical Computer Science**, vol. 21 no. 1, ICGT 2018, May 2019. ISSN 1365-8050. Disponível em: https://dmtcs.episciences.org/4935.

BONDY, A.; MURTY, U. **Graph Theory**. Springer London, 2008. v. 244. (Graduate Texts in Mathematics, v. 244). ISBN 978-1-84628-969-9. Disponível em: https://link.springer.com/book/10.1007/978-1-84628-970-5.

CHARPENTIER, C.; HOCQUARD, H.; SOPENA Éric; ZHU, X. A connected version of the graph coloring game. **Discrete Applied Mathematics**, v. 283, p. 744–750, 2020. ISSN 0166-218X. Disponível em: https://www.sciencedirect.com/science/article/pii/S0166218X20301049.

COSTA, E.; PESSOA, V. L.; SAMPAIO, R.; SOARES, R. Pspace-completeness of two graph coloring games. **Theoretical Computer Science**, v. 824-825, p. 36–45, 2020. ISSN 0304-3975. Disponível em: https://www.sciencedirect.com/science/article/pii/S030439752030181X.

COSTA, E. R. **Resultados em jogos de coloração e perseguição em grafos**. Tese (Tese de Doutorado) – Universidade Federal do Ceará, Fortaleza, CE, Brazil, 2022. Programa de Pós-Graduação em Ciência da Computação.

FORMANOWICZ, P.; TANAŚ, K. A survey of graph coloring - its types, methods and applications. **Foundations of Computing and Decision Sciences**, v. 37, 09 2012.

GARDNER, M. Mathematical games. **Scientific American**, v. 244, n. 4, p. 18–26, 1981.

HAGBERG, A. A.; SCHULT, D. A.; SWART, P. J. **Exploring Network Structure, Dynamics, and Function using NetworkX**. Pasadena, CA USA: [*S. n.*], 2008. 11 - 15 p.

HAVET, F.; ZHU, X. The game Grundy number of graphs. **Journal of Combinatorial Optimization**, Springer Verlag, v. 25, n. 4, p. 752–765, 2013. Disponível em: https://inria.hal.science/hal-00821597.

HUNTER, J. D. Matplotlib: A 2d graphics environment. **Computing in Science & Engineering**, IEEE COMPUTER SOC, v. 9, n. 3, p. 90–95, 2007.

JGRAPH. **draw.io**. 2021. Disponível em: https://www.drawio.com/.

LIMA, C. V.; MARCILON, T.; MARTINS, N.; SAMPAIO, R. Pspace-hardness of variants of the graph coloring game. **Theoretical Computer Science**, v. 909, p. 87–96, 2022. ISSN 0304-3975. Disponível em: https://www.sciencedirect.com/science/article/pii/S0304397522000494.

LIMA, C. V.; MARCILON, T.; MARTINS, N.; SAMPAIO, R. The connected greedy coloring game. **Theoretical Computer Science**, v. 940, p. 1–13, 2023. ISSN 0304-3975. Disponível em: https://www.sciencedirect.com/science/article/pii/S0304397522006454.

Python Software Foundation. **Python Documentation: Data Structures - Dictionaries**. 2024. Acessado em: 08 mar. 2025. Disponível em: https://docs.python.org/3/tutorial/datastructures.html#dictionaries.

RAHMAN, M. L.; WATSON, T. 6-uniform maker-breaker game is pspace-complete. **Combinatorica**, v. 43, p. 595–612, 2023. Disponível em: https://api.semanticscholar.org/CorpusID:220687374.

RUSSELL, S.; NORVIG, P. **Artificial Intelligence: A Modern Approach**. 3rd. ed. USA: Prentice Hall Press, 2009. ISBN 0136042597.

SIDOROWICZ, E. The game chromatic number and the game colouring number of cactuses. **Information Processing Letters**, v. 102, n. 4, p. 147–151, 2007. ISSN 0020-0190. Disponível em: https://www.sciencedirect.com/science/article/pii/S0020019006003541.

SIPSER, M. **Introduction to the Theory of Computation**. 3rd. ed. Cengage Learning, 2012. ISBN 978-1133187790. Disponível em: https://www.amazon.com/Introduction-Theory-Computation-Michael-Sipser/dp/113318779X.

THADANI, S.; BAGORA, S.; SHARMA, A. Applications of graph coloring in various fields. **Materials Today: Proceedings**, v. 66, p. 3498–3501, 2022. ISSN 2214-7853. 3rd International Conference on "Advancement in Nanoelectronics and Communication Technologies"(ICANCT-2022). Disponível em: https://www.sciencedirect.com/science/article/pii/S2214785322044108.

Wikipedia contributors. **Handshaking lemma — Wikipedia, The Free Encyclopedia**. 2024. Acessado em: 10 mar. 2025. Disponível em: https://en.wikipedia.org/w/index.php?title=Handshaking_lemma&oldid=1255260170.

APÊNDICE A – ALGORITMO DA REDUÇÃO DE UMA FÓRMULA POS-CNF PARA UM GRAFO

Código-fonte 1 – Algoritmo em Python da redução de uma fórmula POS-CNF para um grafo

```
def reducao(formula):
    G = nx.Graph()
    G.add_node('V')
    count_clausula = 1
    for clausula in formula["clausulas"]:
        G.add_node(f"C{count_clausula}") # C1, C2, C3, ...
        G.add_edge('V', f"C{count_clausula}") # V - C1, V -
            C2, V - C3, ...
        criar_ciclo_clausula(G, clausula, count_clausula)
        count_clausula += 1
    criar_vertices_variavel(formula, G)
    nx.set_node_attributes(G, colors[0], 'color')
    return G
```

Código-fonte 2 – Código em Python que contém a representação da fórmula POS-CNF na forma de dicionário

```
total_variaveis = 6
pos_cnf_phi = {
    'clausulas': [
        ["X1", "X2", "X2", "X2", "X2", "X1"],
        ["X1", "X2", "X3", "X4", "X5", "X6"]
    ],
    'variaveis': [],
    'valoracao': {
    }
}
pos_cnf_phi["variaveis"] = [f"X{i+1}" for i in range(
```

```
    total_variaveis)] # criando as variaveis
for i in range(total_variaveis):
    pos_cnf_phi["valoracao"][f"X{i+1}"] = None # valoracao
        de cada variavel
```

Código-fonte 3 – Função em Python que cria os *ciclos cláusula* (Seção 5.3.1)

```
def criar_ciclos_clausula(G, clausula, count_clausula):
    for i in range(len(clausula)):
        variavel = f"C{count_clausula}L{i + 1}{clausula[i]}
            " # C1L1X1, C1L2X2, C1L3X3, ...
        G.add_node(variavel)
        if i == 0:
            G.add_edge(f"C{count_clausula}", variavel)
        elif i == len(clausula) - 1:
            G.add_edge(f"C{count_clausula}", variavel)
            G.add_edge(f"C{count_clausula}L{i}{clausula[i -
                1]}", variavel)
        else:
            G.add_edge(f"C{count_clausula}L{i}{clausula[i -
                1]}", variavel)
```

Código-fonte 4 – Função em Python que cria os vértices das variáveis e os ligam com os *vértices cláusula* correspondentes (Seção 5.3.1)

```
def criar_vertices_variavel(formula, G):
    for variavel in formula["variaveis"]:
        G.add_node(variavel) # X1, X2, X3, ...
        G.add_node(f"{variavel}\'") # X1', X2', X3', ...
        G.add_edge(variavel, f"{variavel}\'") # X1 - X1',
            X2 - X2', X3 - X3', ...
        for vertice in G.nodes:
            if variavel in vertice and variavel != vertice:
```

```
8                    G.add_edge(variavel, vertice)
```

APÊNDICE B – ALGORITMO MINIMAX PARA O JOGO DA COLORAÇÃO GULOSA

Código-fonte 5 – Algoritmo Minimax para o Jogo da Coloração Gulosa em Python

```
def minimax(G, k, jogada):
    if grafo_colorido(G, k) == True:
        return True, "AEND", None, None
    if vertice_nao_colorivel(G, k)[0] == True:
        return False, "BEND", None, None
    vertices_nao_coloridos = pegar_vertices_nao_coloridos(G
       )
    if jogada == jogadas["Alice"]:
        for vertice in vertices_nao_coloridos:
            colorir_vertice(G, vertice)
            if minimax(G, k, jogadas["Bob"]) == True:
                descolorir_vertice(G, vertice)
                return True, "A", vertice, G.nodes[vertice
                   ]["color"]
            descolorir_vertice(G, vertice)
        return False, "A", None, None
    elif jogada == jogadas["Bob"]:
        for vertice in vertices_nao_coloridos:
            colorir_vertice(G, vertice)
            if minimax(G, k, jogadas["Alice"]) == False:
                descolorir_vertice(G, vertice)
                return False, "B", vertice, G.nodes[vertice
                   ]["color"]
            descolorir_vertice(G, vertice)
        return True, "B", None, None
```

Código-fonte 6 – Código em Python contendo a representação das cores para serem utilizadas no jogo

```
colors = {
    1: 'red',
    2: 'green',
    3: 'blue',
    4: 'yellow',
    0: 'gray' # a cor cinza representa os vertices nao
       coloridos
}
```

Código-fonte 7 – Código em Python para representar o jogador da vez (Alice ou Bob)

```
jogadas = {
    'Alice': 0,
    'Bob': 1
}
```

Código-fonte 8 – Função em Python que indica se o grafo está totalmente colorido com a quantidade de cores indicada

```
def grafo_colorido(G, qtd_cores):
    for vertice in G.nodes:
        if G.nodes[vertice]["color"] == colors[qtd_cores +
           1] or G.nodes[vertice]["color"] == colors[0]:
            return False
    return True
```

Código-fonte 9 – Função em Python que indica se o grafo possui algum vértice que não pode ser colorido com o número indicado de cores escapechar

```
def vertice_nao_colorivel(G, k):
    for vertice in G.nodes:
        cores = []
```

```
4           vizinhos = list(G.neighbors(vertice))
5           for vizinho in vizinhos:
6               if G.nodes[vizinho]["color"] not in cores and G
                    .nodes[vizinho]["color"] != colors[0]:
7                   cores.append(G.nodes[vizinho]["color"])
8           if len(cores) >= k:
9               return True, vertice
10      return False, None
```

Código-fonte 10 – Função em Python que retorna uma lista com os vértices não coloridos de um grafo escapechar

```
1 def pegar_vertices_nao_coloridos(G):
2     return [vertice for vertice in G.nodes if G.nodes[
          vertice]["color"] == colors[0]]
```

Código-fonte 11 – Função em Python que colore um vértice do grafo com a menor cor possível

```
1 def colorir_vertice(G, vertice):
2     G.nodes[vertice]["color"] = colors[pegar_menor_cor(G,
          vertice)]
```

Código-fonte 12 – Função em Python que retorna a menor cor possível para um vértice

```
1 def pegar_menor_cor(G, vertice):
2     cor = 1
3     while verificar_cores_vizinhas(G, vertice, colors[cor])
          :
4         cor += 1
5     return cor
```

Código-fonte 13 – Função em Python que indica se algum vizinho do vértice está colorido com a cor passada por parâmetro

```python
def verificar_cores_vizinhas(G, vertice, cor):
    vizinhos = list(G.neighbors(vertice))
    for vizinho in vizinhos:
        if cor == G.nodes[vizinho]["color"]:
            return True
    return False
```

Código-fonte 14 – Função em Python que “descolore” um vértice ao atribuir a cor cinza

```python
def descolorir_vertice(G, vertice):
    G.nodes[vertice]["color"] = colors[0] # cinza
```

APÊNDICE C – IMPLEMENTAÇÃO DO JOGO DA COLORAÇÃO GULOSA

Código-fonte 15 – Implementação em Python de uma versão jogável do *Jogo da Coloração Gulosa*

```
def jogo_coloracao_gulosa(G, phi, jogador):
    jogador_escolhido = jogadas[input("Escolha um jogador (Alice ou Bob): ")]
    jogo = 0
    num_turnos = 0
    k = 3
    while(jogo == 0):
        vertices_nao_coloridos = pegar_vertices_nao_coloridos(G)
        print(f"Turno: {num_turnos}")
        if jogador == jogadas["Alice"]:
            if jogador_escolhido == jogadas["Alice"]:
                vertice = input("ALICE - Digite o vertice: ")
                if vertice in vertices_nao_coloridos:
                    colorir_vertice(G, vertice)
                else:
                    print("Vertice ja esta colorido ou nao existe")
                    continue
            else:
                G_copia = G.copy()
                res = minimax(G_copia, 3, jogadas["Alice"])
                if res[0] == False:
                    vertice = pegar_vertices_nao_coloridos(G)[0];
                    colorir_vertice(G, vertice)
```

```
23                      print(f"Alice perde: {res[1]} - Vertice
                            : {vertice} - Cor: {G.nodes[vertice
                            ]['color']}")
24                  else:
25                      colorir_vertice(G, res[2])
26                      print(f"Alice vence: {res[1]} - Vertice
                            : {res[2]} - Cor: {res[3]}")
27              jogador = jogadas["Bob"]
28          else:
29              if jogador_escolhido == jogadas["Bob"]:
30                  vertice = input("Bob - Digite o vertice: ")
31                  if vertice in vertices_nao_coloridos:
32                      colorir_vertice(G, vertice)
33                  else:
34                      print("Vertice ja esta colorido ou nao
                            existe")
35                      continue
36              else:
37                  G_copia = G.copy()
38                  res = minimax(G_copia, 3, jogadas["Bob"])
39                  if res[0] == True:
40                      vertice = pegar_vertices_nao_coloridos(
                            G)[0];
41                      colorir_vertice(G, vertice)
42                      print(f"Bob perde: {res[1]} - Vertice:
                            {vertice} - Cor: {G.nodes[vertice]['
                            color']}")
43                  else:
44                      colorir_vertice(G, res[2])
45                      print(f"Bob vence: {res[1]} - Vertice:
                            {res[2]} - Cor: {res[3]}")
46              jogador = jogadas["Alice"]
```

```
47          if grafo_colorido(G, k):
48              jogo = 1
49          if vertice_nao_colorivel(G, k)[0]:
50              jogo = -1
51              vertice = vertice_nao_colorivel(G, k)[1]
52          num_turnos += 1
53          mostrar_grafo(G, phi)
54      if jogo == 1:
55          print("Jogo terminou com Alice vencendo")
56      else:
57          print(f"Jogo terminou com Bob vencendo - Vertice
              nao colorivel: {vertice}")
```